A Hybrid Quantum-Classical Architecture for Combinatorial Decision Optimization in Networked Systems

by

Huixiang Zhang

B.Sc. Delaware State University, 2024

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE FACULTY OF GRADUATE STUDIES
OF LAKEHEAD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

A Hybrid Quantum-Classical Architecture for Combinatorial Decision Optimization in Networked Systems

by

Huixiang Zhang

Supervisory Committee

Dr. Mahzabeen Emu,
Supervisor
(*Department of Computer Science, Lakehead University, Ontario, Canada*)

Dr. Thiago E. Alves de Oliveira,
Co-Supervisor
(*Department of Computer Science, Lakehead University, Ontario, Canada*)

Dr. Xing Tan,
Internal Examiner
(*Department of Computer Science, Lakehead University, Ontario, Canada*)

Dr. Elif Ak,
External Examiner
(*Department of Electrical and Computer Engineering, Memorial University, Newfoundland and Labrador, Canada*)

ABSTRACT

Combinatorial decision optimization problems arise widely in modern networked systems, where limited communication, computing, and service resources must be efficiently allocated under complex operational constraints. Representative examples include supply-demand matching in data markets, topology control in self-organizing Unmanned Aerial Vehicle (UAV) swarms, and microservice scheduling across the cloud-edge continuum. These problems are typically NP-hard, and as system scale increases or operating conditions evolve rapidly, traditional Mixed-Integer Linear Programming (MILP) formulations often become difficult to solve within real-time decision windows. As a unified binary optimization framework, Quadratic Unconstrained Binary Optimization (QUBO) provides a common way to map diverse combinatorial problems to quantum annealing and quantum-inspired solvers with the potential for significant computational speed advantages. However, the practical use of QUBO in real networked systems still faces three major barriers. First, QUBO modeling remains manual, expert-dependent, and error-prone. Second, standard QUBO formulations are inherently static and therefore not well suited to time-varying environments. Third, the binary representation of QUBO does not naturally align with the continuous resource allocation requirements of real systems. To address these limitations, this thesis develops a hybrid quantum-classical optimization methodology for networked systems. It first formulates and validates domain-specific QUBO models for representative applications. Then it generalizes these efforts through two-stage hybrid frameworks that combine offline combinatorial optimization with lightweight online decision-making for dynamic UAV topology control and robust microservice scheduling. Finally, it investigates large language model driven automation of the MILP-to-QUBO pipeline and integrates Benders decomposition to improve scalability for larger problem instances. Overall, this thesis shows that QUBO can serve not only as a problem-specific solution form, but also as a transferable modeling layer that connects heterogeneous network optimization tasks with near-term quantum hardware, thereby providing a practical pathway toward quantum-enhanced decision-making.

ACKNOWLEDGEMENTS

I would first like to express my deepest gratitude to my supervisor, Dr. Mahzebeen Emu. With exceptional wisdom, patience, and dedication, she guided me onto the right path in academic research. She offered me far more encouragement and support than I could ever have expected, helping me escape endless self doubt. The countless hours she devoted to discussion, thoughtful feedback, and careful refinement of every detail laid the foundation for all that I have achieved academically. With her rich experience and unwavering generosity, she illuminated the road ahead without reservation. It is a great honor to have been her first graduate student, and I will carry her guidance with me as I continue my academic journey.

I would also like to sincerely thank Dr. Faria Khandaker. Through her rigor, patience, and guidance in our collaborative work, she helped me, as a newcomer, complete my first professional research task.

I am sincerely grateful to Dr. Xing Tan and Dr. Elif Ak for kindly agreeing to serve on my committee.

I am deeply thankful to my parents. Your long years of love, sacrifice, and unwavering support have been the foundation upon which I stand.

Finally, I would like to thank Mingxu Yang, whose companionship has been a constant source of courage that carried me forward. I am also grateful to Lianxin Xin, Jinchang Li, Haoxiang Ji, Ruijian Liang, Yuchen Zou, and Yingchen Shi. In different ways, you have all helped shape the person I am today.

PUBLICATIONS

Parts of this thesis have been published or accepted for publication:

- The paper **Coherent Optical Quantum Computing-Aided Resource Optimization for Transportation Digital Twin Construction** [1] was published in the *2025 IEEE International Conference on Collaborative Advances in Software and Computing (CASCON), November 2025.* (part of Chapter 3)
- The paper **LLM-QUBO: An End-to-End Framework for Automated QUBO Transformation from Natural Language Problem Descriptions** [2] was published in the proceedings of the *2025 AAAI Fall Symposium Series.* (part of Chapter 6)
- The paper **Quantum Takes Flight: Two-Stage Resilient Topology Optimization for UAV Networks** [3] has been accepted by the *IEEE International Conference on Communications (ICC) 2026.* (part of Chapter 4)

Contents

Supervisory Committee **ii**

Abstract **iii**

Acknowledgements **iv**

Publications **v**

Table of Contents **vi**

List of Tables **ix**

List of Figures **x**

1 Introduction **1**

1.1 QUBO as a Modeling Abstraction 1

1.2 Why Hybrid Quantum Classical Architecture 3

1.3 Research Gaps and Thesis Questions 4

1.4 Thesis Overview and Main Contributions 5

1.5 Organization of Thesis 7

2 Literature Review and Background Notions **9**

2.1 Combinatorial Decision Optimization in Networked Systems 9

2.2 Classical mathematical programming foundations 11

2.3 QUBO as an intermediate abstraction layer 12

2.4 Quantum annealing and hardware-aware execution 14

2.5 Hybrid quantum-classical optimization architecture 17

2.6 Synthesis of literature and thesis gaps 20

3 Stochastic Resource Planning for Metaverse Data Marketplaces **22**

3.1 Introduction 22
3.2 System Model 24
3.3 QUBO Transformation 27
3.3.1 Binary Encoding of Integer Variables 27
3.3.2 Penalty-Based Constraint Embedding 27
3.4 Experimental Evaluation 29
3.4.1 Solution Quality and Computation Time 30
3.4.2 Analysis of Encoding and Scalability 31
3.5 Summary 32

4 Two-Stage Resilient Topology Control for UAV Communication Networks 33
4.1 Introduction 33
4.2 Related Work 36
4.3 System Model and Proposed Framework 37
4.3.1 Network Model and Objective Function 38
4.3.2 Stage 1: Offline Strategic Computation 39
4.3.3 Stage 2: Real-time Evaluation and Selection 40
4.3.4 Compatible Deployment with SDN/O-RAN 41
4.4 Experimental Evaluation 42
4.4.1 Solution Quality and Diversity Comparison 42
4.4.2 Quantum versus Classical Solver Performance 43
4.4.3 Performance-Fragility Trade-off Analysis 44
4.4.4 Dynamic Performance Evaluation 45
4.5 Summary 45

5 Adaptive Microservice Chain Scheduling in the Cloud-Edge Continuum 46
5.1 Introduction 46
5.2 Related Work 49
5.3 System Model 50
5.3.1 Infrastructure and System State 51
5.3.2 Control Action and Surrogate Objective 52
5.4 Quantum-Inspired and Adaptive Robust Scheduling Framework . . . 53
5.4.1 State Extraction and Dimensionality Reduction 54

5.4.2 QUBO Formulation and Rank Generation 54
5.4.3 Continuous Computing Resource Allocation 56
5.4.4 Adaptive Robust Execution and Safety Guarantee 57
5.5 Experimental Evaluation 59
5.5.1 Performance Gain from Structural Prior 60
5.5.2 Resilience Under Stochastic Volatility 61
5.5.3 System Dynamics and Shock Recovery 62
5.6 Summary 63

6 Automated QUBO Formulation and Hybrid Orchestration for Combinatorial Optimization 64
6.1 Introduction 65
6.2 Related Work 67
6.3 Framework Design 69
6.3.1 Overall Architecture and Workflow 69
6.3.2 QUBO Transformation Engine 70
6.3.3 Hybrid Benders Decomposition 71
6.4 Experimental Evaluation 72
6.4.1 QUBO Conversion Correctness 72
6.4.2 Scalability of Hybrid Benders Decomposition 74
6.5 Summary 76

7 Conclusions 78
7.1 Limitations and Future Research Directions 79
7.1.1 Beyond the centralized full-information assumption 79
7.1.2 Incorporating decision-aware learning into QUBO parameterization 80
7.1.3 Toward online and adaptive hybrid orchestration 81
7.1.4 Toward deployment-level evaluation 83
7.2 Final Remarks 83

A List of Abbreviations 84

Bibliography 87

List of Tables

Table 1.1	Summary of chapters and their methodological progression	7
Table 2.1	Mapping of literature foundations to research gaps and thesis chapters.	21
Table 3.1	Parameters and Decision Variables for Resource Subscription Model	25
Table 3.2	Parameter Configurations Across Problem Scales	30
Table 3.3	Execution Time Comparison Across Problem Scales	31
Table 4.1	List of Key Notations Used in the Proposed Framework.	37
Table 5.1	Summary of Core Sets, States, and Decision Variables	51
Table 6.1	Analysis of automated MILP-to-QUBO conversion correctness for nine classical optimization problems.	73

List of Figures

Figure 1.1 Representative combinatorial decision optimization scenarios in networked systems, including metaverse data marketplace matching, UAV communication topology control, and cloud-edge microservice chaining scheduling. 2

Figure 1.2 From classical optimization models to hybrid solution workflows through QUBO abstraction. 3

Figure 1.3 Key barriers between QUBO formulation and deployable hybrid optimization in networked systems. 4

Figure 2.1 Methodological pipeline from classical formulations to QUBO abstraction and hybrid execution paths. . . . 11

Figure 3.1 The workflow of Metaverse Service Provider with two-phase training data. 23

Figure 3.2 Comparison of logarithmic total costs obtained via CIM (quantum hardware), Gurobi (QUBO mode), and `dwave-neal` simulated annealing under small, medium, and large problem configurations defined in Table 3.2 31

Figure 3.3 Comparison of computation time between the coherent Ising machine, left axis, and Gurobi, right axis, for solving the target QUBO formulation across small, medium, and large problem scales. 32

Figure 4.1 Two-stage topology control framework: offline generation of diverse candidate topologies via QA (Steps 1-2), and online lightweight selection based on real-time network conditions (Step 3). 35

Figure 4.2 Comparison of QA and SA in Stage 1: (a) solution quality and diversity across five traffic centralization scenarios; (b) runtime scaling with network size. . . . 42
Figure 4.3 Offline parameter selection and dynamic performance: (a) throughput versus load balance trade-off as β varies; (b) performance retention comparison between the single optimization model and the dynamic two-stage framework. 44

Figure 5.1 End-to-end request paths and node-local scheduling bottlenecks in a microservice-based user-facing application. 47
Figure 5.2 Overview of the closed-loop adaptive scheduling framework integrating SQA and real-time feedback control. 54
Figure 5.3 (a) Histogram showing the distribution of relative performance improvements over the greedy baseline. (b) Scatter plot comparing solution quality, indicating consistent improvements in high-cost regions. 60
Figure 5.4 (a) Impact of increasing uncertainty level α on average weighted completion time. (b) Cumulative Distribution Function (CDF) of completion times under high volatility ($\alpha = 1.5$), highlighting the mitigation of heavy tail risks. 61
Figure 5.5 System dynamics under a simulated shock interval from time step 300 to 900. (a) The cross-system mean trust parameter $\bar{\lambda}(t)$ drops to trigger the safeguard mode. (b) Q-GARS effectively suppresses the peak surge of the 95th percentile queue backlog. (c) Blocked capacity ratio is minimized and recovers to near-zero levels fastest. 62

Figure 6.1 Overview of the automated QUBO formulation and hybrid orchestration framework 69
Figure 6.2 Convergence Failure of the Monolithic QUBO Approach on a 20×20 CFLP Instance. 74

Figure 6.3 Scalability Comparison of the Hybrid Benders Decomposition against a Direct MILP Solver. 75
Figure 6.4 Overview of the convergence behavior in the CFLP Benders decomposition approach. 76
Figure 6.5 Evolution of Objective Function Components During Benders Decomposition. 77

Chapter 1

Introduction

Many core management tasks in modern networked systems can be understood as combinatorial decision optimization under limited resources, dynamic demand, and latency constraints. Representative scenarios include supply-demand matching in data markets, topology maintenance in Unmanned Aerial Vehicle (UAV) swarms, and computation and microservice chain scheduling across the cloud-edge continuum, as illustrated in Fig. 1.1 [4–6]. These problems require the coordinated handling of multiple coupled constraints and objectives over discrete decision spaces, and therefore their formulation and solution quickly evolve into high-complexity combinatorial optimization tasks [3]. Although classical mathematical programming methods, especially Mixed-Integer Linear Programming (MILP), have long served as important tools for solving such problems, relying only on traditional exact optimization pipelines often makes it difficult to balance solution quality, solving time, and online adaptability when the system scale continues to grow, environmental conditions change frequently, and decisions must be made within tight time windows [7,8]. This motivates the need for a more scalable optimization paradigm that can provide a unified modeling layer between complex discrete decision problems and heterogeneous computational backends for efficient decision-making in networked systems.

1.1 QUBO as a Modeling Abstraction

In this context, Quadratic Unconstrained Binary Optimization (QUBO) provides a unified modeling interface for complex discrete decision problems in networked systems. By compressing selection variables, interaction terms, and constraint penalties

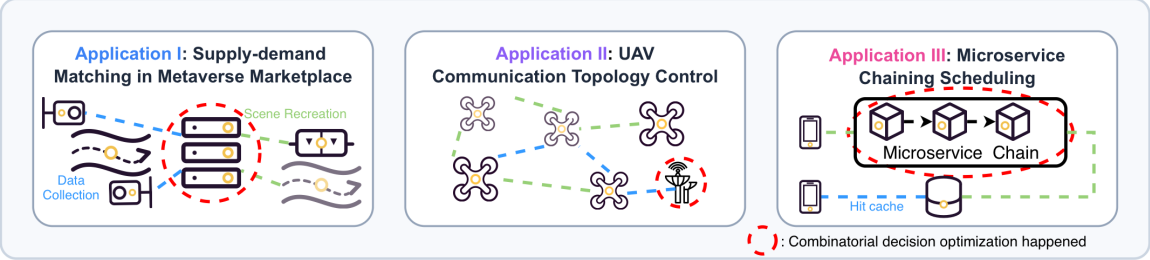


Figure 1.1: Representative combinatorial decision optimization scenarios in networked systems, including metaverse data marketplace matching, UAV communication topology control, and cloud-edge microservice chaining scheduling.

into a single quadratic binary energy function, it offers a shared representation for quantum annealers, quantum inspired annealers, and selected classical and hybrid optimization workflows [9, 10]. Many NP hard combinatorial optimization problems can be transformed into the Ising model through variable encoding and penalty construction [11]. Since the Ising model is mathematically equivalent to QUBO, this establishes QUBO as a standard modeling language for such complex problems [12]. However, this unification does not imply that practical deployment is straightforward. In real systems, constraints usually need to be explicitly embedded in the objective through penalty terms [13]. Currently, higher order relations, dense variable interactions, and hardware connectivity limitations introduce additional overhead in quadratization, embedding, and decomposition [14]. Therefore, QUBO is better understood as an intermediate abstraction layer connecting application level modeling to hybrid solution architectures, rather than as a universal endpoint operating independently of problem semantics and computational backend [2, 15].

Figure 1.2 summarizes how the thesis positions QUBO within the overall solution pipeline. Rather than being executed directly on quantum hardware, classical optimization models are first transformed through binarization, penalty reformulation, and variable encoding into a QUBO representation that is compatible with binary quadratic optimization workflows [2]. In this sense, QUBO should not be understood as the final solution method but as an intermediate abstraction layer that bridges heterogeneous classical formulations and downstream hybrid solution mechanisms. Once a problem has been expressed in this common form, the same QUBO instance can be routed to different computational workflows according to the problem scale, structural properties, and hardware constraints.

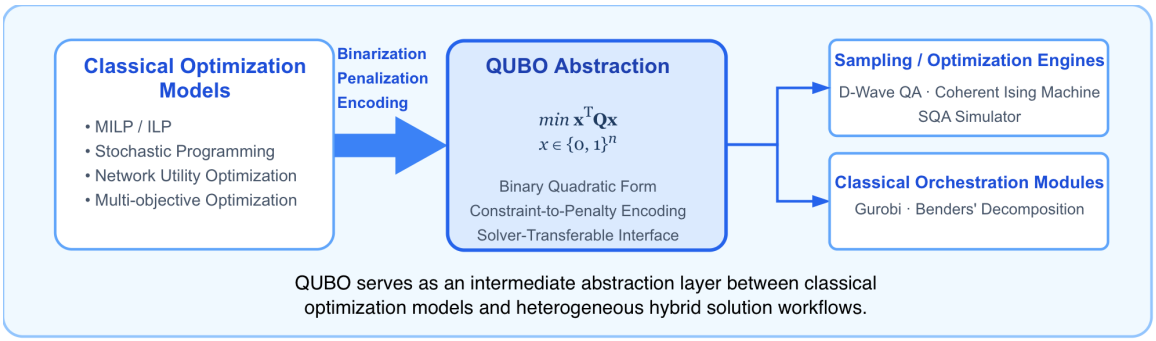


Figure 1.2: From classical optimization models to hybrid solution workflows through QUBO abstraction.

1.2 Why Hybrid Quantum Classical Architecture

However, reformulating an optimization problem as QUBO does not automatically translate into scalable practical solvability. Because near-term quantum hardware remains constrained by qubit count, connectivity, noise, and embedding overhead, making purely quantum workflows difficult to apply directly to large and dense real-world instances [16, 17]. A more realistic path is not to completely replace classical optimization with quantum computation, but to construct a hybrid quantum-classical architecture [8]. In this design, problem decomposition, constraint management, parameter tuning, and global search control remain on the classical side, whereas highly correlated, strongly non-convex, or locally expensive subproblems are delegated to quantum or quantum-inspired modules [9, 18]. Within such an architecture, the classical solver preserves the overall stability of the optimization process through feasibility checks, bound updates, node selection, and post-processing, whereas the quantum component is better positioned as a sampler, heuristic callback, or accelerator for selected subproblems [18, 19].

Existing studies suggest that this division of labor is especially important for solving QUBO under current hardware conditions. It preserves the interpretability and control of classical methods while mitigating the performance bottlenecks of purely quantum approaches on large-scale instances [10]. In this thesis, hybrid is therefore treated as an architectural principle rather than a simple concatenation of solvers, with the aim of establishing a scalable and transferable co-optimization mechanism across heterogeneous computational backends for practical networked systems.

1.3 Research Gaps and Thesis Questions

Despite the growing interest in QUBO-based and hybrid quantum-classical optimization, most existing studies remain at the stage of validating individual problems. Four research gaps remain between the current state of the art and a transferable, scalable, and deployable optimization architecture for networked systems, as summarized in Fig. 1.3.

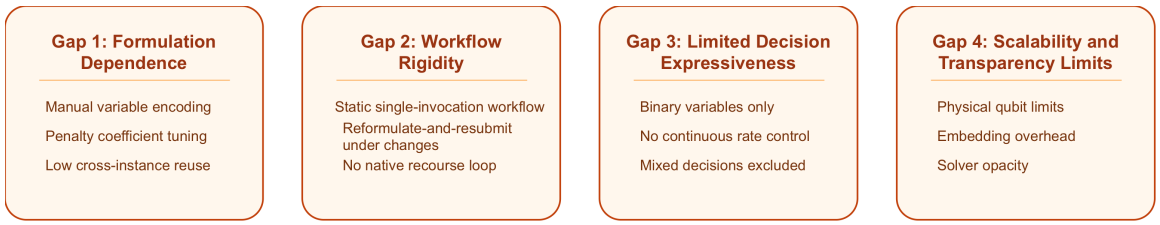


Figure 1.3: Key barriers between QUBO formulation and deployable hybrid optimization in networked systems.

The first gap is the formulation dependence. Existing QUBO workflows still rely heavily on manual variable encoding, constraint penalization, penalty calibration, and formulation-specific trade-offs, which limits reuse and complicates integration into conventional optimization pipelines [12, 13, 15].The second gap is better characterized as workflow rigidity rather than a formal limitation of the QUBO mathematical model itself. In practice, many QUBO-based pipelines are organized around a fixed submitted instance, so when demand, topology, or operating conditions change, the optimization problem often needs to be regenerated or resubmitted, making real-time recourse and continuous adaptation difficult to support naturally [7, 19].

The third gap is limited decision expression. Because QUBO is defined over binary variables, continuous allocation, rate control, and mixed discrete-continuous decisions must typically be represented indirectly through additional encoding, which increases modeling complexity and weakens the natural alignment between the optimization model and the real resource-allocation processes [20, 21]. The fourth gap is scalability and transparency limits. Even after a valid QUBO has been obtained, dense couplings, penalty-induced connectivity, and sparse hardware topologies can substantially increase minor-embedding cost, physical qubit usage, and chain-break risk, while black-box hybrid solvers restrict fine-grained user control over decomposition, tuning, and runtime behavior [8, 10, 16, 22]. Taken together, these four gaps indicate that the central challenge of this thesis is not merely to improve a single solver, but to close the broader methodological distance between modelability and

deployability in practical networked optimization systems.

Based on the four research gaps summarized in Fig. 1.3, this thesis formulates four connected thesis questions. The first asks how the dependence of QUBO modeling on manual expertise can be reduced, so that a reusable and automated formulation pipeline can be established. The second asks how a QUBO-based optimization workflow can better respond to changing system states, rather than remaining tied to a static single-instance solving pattern. The third asks how the discrete strengths of QUBO can be preserved while incorporating continuous resource-allocation processes, so that mixed decision structures in real systems can be modeled more naturally. The fourth asks how a transparent and scalable hybrid quantum-classical architecture can be constructed under current hardware constraints, with a view toward stable deployment in practical networked systems. The following Thesis Overview and Main Contributions section explains how each question is addressed through the methodological progression of the thesis.

1.4 Thesis Overview and Main Contributions

Based on the research gaps and thesis questions established above, the core contribution of this thesis does not lie in proposing a new QUBO formulation for a single optimization task in modern network systems. Instead, it develops a methodological progression from problem modeling to practical deployment.

As summarized in Table 1.1, this progression consists of four connected stages. The first stage focuses on domain-grounded formulation and empirical validation. Its goal is to verify how classical optimization models, such as Stochastic Integer Programs (SIP), can be transformed into QUBO in a stable manner and connected to a specific quantum or quantum-inspired backend [1]. The second stage focuses on recourse-aware dynamic adaptation. Its goal is to move beyond static single-instance workflows so that a QUBO-based optimization framework can better respond to time-varying environments [3]. The third stage advances toward discrete-continuous co-design, where binary combinatorial search is coordinated with continuous resource allocation [23]. And the fourth stage addresses automated formulation and scalable hybrid orchestration by integrating formulation automation, decomposition, and transparent hybrid solving into a unified architecture [2].

The first two stages answer how QUBO-based optimization can move from modelability to validation, and then from validation to adaptation. In Stage 1, the thesis

begins with a networked resource planning problem under uncertain demand. It transforms a two-stage SIP model into a QUBO, maps the resulting QUBO matrix to a coherent Ising machine (CIM), and compares the results with Gurobi and simulated annealing (SA) using dwave-neal on the same QUBO matrix [24–26]. The significance of this stage is that it demonstrates both the practical operability of QUBO as an intermediate abstraction layer and the feasibility of the classical-model-to-QUBO-to-hardware pipeline in a real application setting. Stage 2 then turns to online adaptation in dynamic network environments. Motivated by the fragility of a single static optimal topology under time-varying link conditions, the thesis develops a two-stage quantum-assisted workflow. The offline stage generates multiple high-quality and structurally diverse candidate topologies by triggering the multi-sampling ability of the QA hardware, while the online stage performs lightweight selection according to the real-time link states [3].

The latter two stages push the thesis methodology toward a more complete hybrid optimization architecture. Stage 3 focuses on discrete-continuous co-design. It no longer treats QUBO as an isolated discrete solver. Instead, it places discrete ranking and continuous resource allocation within the same decision loop. In this stage, the thesis first uses QUBO to generate discrete scheduling priorities for microservice instances [20]. It then applies a continuous allocation mechanism to determine rate allocation under capacity constraints, while an adaptive safeguard handles prediction failure and runtime volatility [27]. This design preserves the combinatorial search strength of QUBO. At the same time, it incorporates the continuous control processes that are unavoidable in real systems [23]. Stage 4 elevates this idea to the level of formulation automation and system orchestration. There are several studies on translating the description of problems in the natural language to a formal optimization model [28]. Accordingly, this thesis focuses on automated QUBO formulated with appropriate penalty coefficient and binarized precision that can be fully compatible with the target quantum hardware. Also, with Benders decomposition integration, this design can deal with a large-scale problem by separating the binary master problem from the continuous subproblem. As a result, QUBO is no longer treated as a monolithic endpoint. It becomes an intermediate representation that supports iterative, scalable, and solver-aware hybrid optimization [2]. This final stage unifies the lessons of the earlier stages into a deployable hybrid orchestration framework, moving the thesis from problem-level modeling toward architecture-level integration.

1.5 Organization of Thesis

The remainder of the chapters are organized as follows.

Chapter 2 reviews the literature and discusses the background topics relevant to the thesis.

The remaining chapters are organized according to a methodological progression from problem modeling to deployable hybrid optimization architecture. Table 1.1 summarizes this progression by mapping each chapter to its representative networked system, methodological stage, and primary research gap. Rather than presenting the chapters as a set of parallel application cases, the thesis emphasizes the methodological connection among them. More specifically, it shows how the research develops from domain-specific QUBO formulation and validation toward dynamic adaptation, discrete-continuous coordination, and automated formulation with scalable hybrid orchestration. This organization allows the later chapters to function not only as individual studies, but also as interconnected steps toward the unified hybrid quantum-classical architecture proposed in this thesis.

Table 1.1: Summary of chapters and their methodological progression

Chapter	Representative System	Methodological Stage	Research Gap Addressed
3	Metaverse data marketplace	Domain-grounded formulation and empirical validation	G1: Formulation Dependence
4	UAV swarm topology control	Two-stage recourse-aware dynamic adaptation	G2: Workflow Rigidity
5	Cloud-edge microservice scheduling	Discrete–continuous co-design	G3: Limited Decision Expressiveness
6	General combinatorial optimization pipeline	LLM-driven automated formulation and hybrid orchestration	G1 + G4: Formulation Dependence and Scalability

More specifically, Chapter 3 begins with a stochastic resource planning problem in a metaverse data marketplace. It establishes a conversion pipeline from two-stage stochastic integer programming to QUBO and empirically validates the resulting formulation on practical computational backends. Chapter 4 then extends this line of work to dynamic UAV communication topology control. By separating offline candidate generation from online lightweight selection, it addresses the limitation of

fixed single-instance workflows under time-varying conditions. Chapter 5 turns to microservice chain scheduling across the cloud-edge continuum. It develops a mixed decision architecture in which discrete ranking and continuous resource allocation are coordinated, thereby addressing the expressive gap between binary QUBO modeling and continuous control requirements in real systems. Building on these results, Chapter 6 abstracts the earlier domain-specific designs into a higher-level layer of automated formulation and hybrid orchestration. Through an LLM-driven formulation pipeline and decomposition-based hybrid solving, it unifies the lessons of the preceding chapters into a more transferable and scalable optimization framework. Taken together, the thesis chapters define a clear methodological trajectory, moving from domain-grounded formulation and empirical validation to recourse-aware adaptation, discrete-continuous co-design, and deployable hybrid orchestration.

Chapter 2

Literature Review and Background Notions

This chapter outlines the core theoretical and methodological concepts required to understand the foundation of this thesis, organized around the central theme of combinatorial decision optimization in networked systems. Specifically, the chapter first identifies the common structural features of networked optimization problems in terms of decision variables, coupled constraints, and dynamic operating conditions. It then introduces the classical optimization foundations most relevant to this thesis and explains why QUBO is better understood as an intermediate abstraction layer that connects application-level modeling with downstream solution mechanisms, rather than as an isolated final solution form. On this basis, the chapter discusses the hardware execution constraints of quantum annealing, the cost of embedding, and their implications for practical deployment. It also reviews the main developments in hybrid quantum-classical optimization workflows with respect to scalability, controllability, and interpretability.

2.1 Combinatorial Decision Optimization in Networked Systems

Although resource allocation in the metaverse, topology control in UAV swarms, and microservice scheduling across the cloud-edge continuum differ substantially in application semantics, they exhibit highly consistent structural characteristics at the optimization level. First, these problems are commonly built on discrete decisions such as

activation, assignment, matching, ranking, routing, or topology selection, and their feasible sets therefore have an inherently combinatorial nature [29]. Second, communication, computation, storage, energy, and connectivity resources are strongly coupled, so the resulting constraint systems often combine capacity limits, dependency relations, interference relations, and quality-of-service requirements [30–32]. Third, demand fluctuations, link variations, and local failures require many of these problems to be solved under uncertainty and time-varying operating conditions. For this reason, resource management tasks in different networked systems can often be understood in a unified mathematical sense as high-dimensional combinatorial decision optimization problems governed by multiple families of coupled constraints [33]. This shared structure provides the foundation for the classical modeling frameworks introduced in the next section.

The structural properties identified above are significant because they jointly determine the feasibility of a unified modeling pathway. In many networked systems, the core decisions can be reduced to whether a service is deployed, whether a task is assigned to a specific node, or whether a link or topological connection is activated, which allows them to be naturally encoded as discrete decision vectors [34, 35].At the same time, constraints associated with computation, bandwidth, storage, energy, and latency are rarely independent. Instead, they are coupled through resource contention, link dependencies, and service ordering requirements. This coupling means that the constraints cannot be straightforwardly decomposed into independent subproblems, and must instead be expressed through interaction terms among decision variables in any unified formulation [36].

Under dynamic operating conditions, variations in demand, network perturbations, and updates in node availability further require the solution framework to support fast incremental adjustment of existing decisions without reconstructing the entire problem instance from scratch [37]. Because of this continuous structure, spanning discrete decisions, coupled constraints, and dynamic re-decision requirements, many resource management problems in networked systems admit a common methodological basis for unified formulation, reformulation, and subsequent mapping into binary optimization models. This logic also provides a direct transition to the classical mathematical programming frameworks discussed in the next section.

2.2 Classical mathematical programming foundations

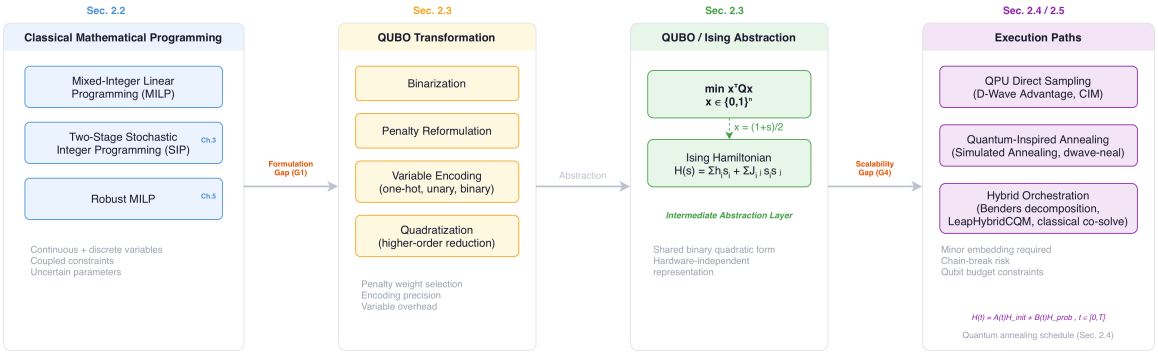


Figure 2.1: Methodological pipeline from classical formulations to QUBO abstraction and hybrid execution paths.

Resource management problems in networked systems involve discrete structural decisions, including deployment, assignment, and link activation, alongside continuous variables governing bandwidth, computation, storage, and latency. As shown in the leftmost column of Fig. 2.1, Mixed-integer linear programming (MILP) therefore serves as the natural classical modeling starting point for this section.

$$\begin{aligned} \min_{x,y} \quad & c^\top x + d^\top y \\ \text{s.t.} \quad & Ex + Fy = e, \\ & Ax + By \le b, \\ & \ell_x \le x \le u_x, \\ & x \in \mathbb{R}^n,\ y \in \{0,1\}^p, \end{aligned} \tag{2.1}$$

where x denotes continuous resource allocation variables and y denotes discrete structural decision variables, with $c \in \mathbb{R}^n$, $d \in \mathbb{R}^p$, $\ell_x, u_x \in \mathbb{R}^n$, $E \in \mathbb{R}^{r\times n}$, $F \in \mathbb{R}^{r\times p}$, $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times p}$, $e \in \mathbb{R}^r$, and $b \in \mathbb{R}^m$ [38].

This form captures the coupling between continuous resource allocation and combinatorial structural decisions within a single objective and constraint system. It therefore serves as the classical modeling starting point for uncertainty extensions, QUBO reformulations, and hybrid solution workflows developed in subsequent sections.

Compared with the deterministic MILP form above, many key parameters in

networked systems are only revealed after part of the decision has already been fixed. It is therefore necessary to introduce a two stage stochastic programming model with recourse:

$$\min_{x \in X} \quad c^\top x + \mathbb{E}_\xi[Q(x, \xi)] \tag{2.2}$$

where

$$Q(x, \xi) = \min_{y(\xi) \in Y(\xi)} \quad q(\xi)^\top y(\xi) \quad \text{s.t.} \quad W(\xi) y(\xi) \geq h(\xi) - T(\xi) x. \tag{2.3}$$

Here, $x \in \mathbb{R}^n$ denotes the first-stage decisions made before the realization of the random parameter ξ, while $y(\xi) \in \mathbb{R}^m$ denotes the second-stage corrective decisions taken after scenario ξ is observed [39]. The coefficients satisfy $c \in \mathbb{R}^n$, $q(\xi) \in \mathbb{R}^m$, $W(\xi) \in \mathbb{R}^{r \times m}$, $T(\xi) \in \mathbb{R}^{r \times n}$, and $h(\xi) \in \mathbb{R}^r$, so the objective and constraint terms remain dimensionally consistent. The significance of this formulation is not merely that it averages over scenarios, but that it explicitly separates here-and-now decisions from corrective actions taken after the uncertain parameter is revealed. In practical networked systems, both stages may involve integer decision variables, such as binary activation or assignment choices, which makes the resulting problem a stochastic mixed-integer program with substantially higher computational complexity than the continuous case. Because both stages may contain integer variables, converting such a formulation into QUBO requires binary encoding across all decision variables in both stages [12]. This observation provides a direct motivation for the QUBO reformulation techniques discussed in the next section.

Beyond scenario-based stochastic modeling, uncertainty can also be described through sets. When probability distributions are unreliable or unavailable, robust optimization requires every constraint to remain feasible for all possible perturbations within a predefined uncertainty set [40]. This treatment extends the representation of uncertainty from scenario enumeration to set-based constraints, and provides a complementary modeling perspective for the discussions in later chapters that involve optimization under uncertain operating conditions.

2.3 QUBO as an intermediate abstraction layer

As shown in the two middle columns of Fig. 2.1, at the classical mathematical programming level developed in the previous section, this connection can be achieved by reformulating heterogeneous constrained optimization models into a common QUBO representation that is compatible with quantum annealing, quantum inspired anneal-

ing, and related hybrid optimization workflows.

For this purpose, QUBO can be written in the compact form

$$\min_{x\in\{0,1\}^n} x^\top Qx, \qquad Q \in \mathbb{R}^{n\times n}, \tag{2.4}$$

where x is a binary decision vector and Q is the coefficient matrix, so that $x^\top Qx$ is a scalar objective function [41]. Without loss of generality, Q can be taken to be symmetric or equivalently upper triangular, since only the sum $Q_{ij} + Q_{ji}$ affects the objective value [12].

If the original model is written as $x^\top Qx+c^\top x$, the linear term $c^\top x$ can be absorbed into the diagonal of Q because binary variables satisfy $x_i^2 = x_i$, yielding an equivalent compact quadratic form $x^\top(Q + \mathrm{diag}(c))x$.

The significance of this representation is not only that it encodes linear costs and pairwise interactions within a single quadratic objective, but also that it provides a shared binary abstraction for optimization models originating from different classical formulations. Accordingly, in this thesis, QUBO is not treated as a universal endpoint detached from problem structure. Instead, it is regarded as an intermediate abstraction layer that connects classical mathematical programming formulations with downstream annealing oriented and hybrid execution workflows.

After the standard QUBO form is established, the more important issue is how feasibility conditions from the original constrained model can be embedded into an unconstrained binary objective. For a general constrained binary optimization model written as

$$\min_{x\in\{0,1\}^n} \quad f(x) \qquad \text{s.t.} \quad Ax = b, \tag{2.5}$$

a penalized QUBO form can be constructed by introducing a penalty parameter $\lambda > 0$:

$$\min_{x\in\{0,1\}^n} \quad f(x) + \lambda\|Ax - b\|_2^2, \tag{2.6}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, so that $Ax - b \in \mathbb{R}^m$ and the term $\|Ax - b\|_2^2$ is a scalar measure of constraint violation whose contribution to the objective is scaled by λ [12]. When λ is chosen large enough, solutions that violate the constraints are penalized relative to feasible ones, allowing the constrained problem to be rewritten as an unconstrained binary model. However, selecting λ is itself a nontrivial modeling step: values that are too small may fail to enforce feasibility, whereas excessively

large values can distort coefficient scales and make the resulting QUBO harder to solve efficiently [42, 43].

For inequality constraints, the standard treatment is to first introduce slack variables to convert them into equalities, and then represent those slack variables through binary expansion when a binary QUBO representation is required. Likewise, if the original decision variables are general integers rather than binary variables, they must also be encoded through binary expansion, which further increases the size of the resulting QUBO. Therefore, the essence of QUBO reformulation is not merely to express an objective as a quadratic form, but to systematically absorb classical constraint structures into a unified binary quadratic representation through penalty design, slack introduction, and binary encoding of nonbinary quantities.

Under a physics-based representation, the Ising model equivalent to QUBO can be written as

$$H(s) = \sum_{i=1}^{n} h_i s_i + \sum_{i<j} J_{ij} s_i s_j, \qquad s_i \in \{-1, +1\}, \tag{2.7}$$

where h_i denotes the local bias acting on spin i and J_{ij} denotes the coupling strength between spins i and j, corresponding respectively to the diagonal and off-diagonal entries of the QUBO matrix Q after variable transformation [11]. By the variable transformation

$$x_i = \frac{1 + s_i}{2}, \tag{2.8}$$

Ising spin variables and binary decision variables are placed in one-to-one correspondence, which allows an Ising Hamiltonian to be converted into a QUBO objective and vise versa. This equivalence ensures that any optimization problem expressed in QUBO form can be directly mapped onto the physical energy landscape of an annealing processor. Together with the penalty-based reformulation techniques introduced above, the QUBO-Ising correspondence therefore completes the modeling chain from classical constrained formulations to hardware-executable energy functions.

2.4 Quantum annealing and hardware-aware execution

After establishing QUBO as an intermediate abstraction layer in the previous section, it is necessary to explain how this binary quadratic model enters a physical hardware execution pipeline [44]. Quantum annealing is an analog optimization method whose

basic mechanism is to evolve a quantum system over a total runtime T according to the time-dependent Hamiltonian

$$\mathcal{H}(t) = A(t)\,\mathcal{H}_{\text{init}} + B(t)\,\mathcal{H}_{\text{prob}}, \qquad t \in [0, T], \tag{2.9}$$

where $A(t)$ decreases from a dominant initial value to near zero and $B(t)$ increases from near zero to a dominant final value over the course of the anneal [45]. The initial Hamiltonian $\mathcal{H}_{\text{init}}$ prepares the system in an easily constructed ground state, while the problem Hamiltonian $\mathcal{H}_{\text{prob}}$ encodes the target optimization instance in Ising or equivalent QUBO form. The adiabatic theorem of quantum mechanics states that if the evolution is sufficiently slow, the system remains in the instantaneous ground state throughout the process, so the final measurement yields the optimal solution of the encoded problem [10]. In practice, however, finite annealing time, qubit noise, and hardware connectivity constraints cause the process to deviate from ideal adiabatic evolution. For this reason, the rest of this section focuses on the relationship among hardware topology, embedding overhead, and practical execution behavior.

A logical QUBO instance cannot generally be executed on quantum annealing hardware in its original form, because the physical qubit connectivity of an actual Quantum Processing Unit (QPU) is sparse. For example, the Chimera topology used in earlier D-Wave systems provides a qubit degree of approximately 6, the Pegasus topology in the Advantage system increases this to 15, and the Zephyr topology in Advantage2 further improves connectivity [8, 46]. For an arbitrarily structured binary quadratic model whose interaction graph may be denser than the hardware graph, the logical variables must first be mapped onto the physical QPU topology through a preprocessing step known as minor embedding [46]. In this process, a single logical variable is often represented by a chain of multiple physical qubits to compensate for missing native couplings. The direct consequence is that embedding increases physical qubit consumption and introduces additional execution burdens, including chain breaks, parameter rescaling, and decoding overhead [10]. Therefore, the practical behavior of quantum annealing is shaped not only by the abstract QUBO formulation, but also by the hardware topology, the size of the working graph, and the quality of the embedding. This also explains why the same logical model can exhibit substantially different embeddable sizes and solution quality across different QPU generations.

Furthermore, the practical performance of quantum annealing cannot be judged

solely by the nominal anneal time, because real execution involves repeated read-anneal cycles, chain-repair procedures, parameter scaling, and postprocessing [10]. Under ideal adiabatic conditions, the system is guaranteed to reach the ground state, but on real hardware the process is better understood as a heuristic sampler that returns a distribution of low-energy solutions rather than a deterministic guaranty of global optimality [47]. This sampling behavior gives rise to a property that has received growing attention in the optimization literature: solution diversity. In simulated annealing, the system explores the solution space by thermally surmounting energy barriers, a sequential process that tends to converge repeatedly to similar locally optimal regions. Quantum annealing instead traverses energy barriers via quantum tunneling rather than thermal fluctuations, providing a different mechanism for escaping local optima. More broadly, classical stochastic solvers based on Monte Carlo techniques suffer from a lack of ergodicity and are prone to mode collapse, where sampling concentrates within a few regions of the solution space [48]. In contrast, the sampling behavior of QA is structurally different and can facilitate the generation of more diverse sets of near-optimal solutions within the same computational time budget. Experimental evaluation on D-Wave processors has shown that QA achieves competitive or superior diversity sampling speed compared to state-of-the-art classical solvers across multiple classes of synthetic problems [49]. This property is directly relevant to application scenarios that require multiple structurally distinct high-quality solutions from a single optimization campaign, and subsequent chapters will exploit it in specific networked optimization contexts.

Recent benchmarking studies indicate that hybrid quantum-classical configurations, which combine QPU sampling with classical postprocessing and decomposition, can achieve higher accuracy and substantially faster problem-solving times than either pure quantum or pure classical approaches on large and dense QUBO instances [10]. These observations reinforce the view that hardware-aware execution is not merely a matter of submitting a QUBO matrix to a QPU, but involves treating embedding, sampling, chain repair, and solution decoding as an integrated pipeline. This understanding prepares the transition to the hybrid quantum-classical orchestration discussed in the next section.

2.5 Hybrid quantum-classical optimization architecture

The discussion in the previous section has shown that limited qubit counts, sparse hardware topologies, and embedding overhead collectively restrict the direct applicability of pure quantum annealing to large and dense QUBO instances. Under near term quantum hardware conditions, purely quantum workflows cannot reliably solve practical optimization problems that exceed the embeddable scale of a single QPU [16]. A common practical view in the current quantum annealing literature is therefore not to replace classical optimization entirely with quantum computation, but to construct a hybrid quantum-classical architecture in which problem decomposition, constraint management, parameter tuning, and global search control remain on the classical side, while highly correlated, strongly non-convex, or locally expensive subproblems are delegated to quantum or quantum-inspired modules [50, 51]. Within this design, the classical solver maintains overall stability through feasibility checking, bound updates, node selection, and postprocessing, while the quantum component is better positioned as a sampler, heuristic callback, or accelerator for selected subproblems [19]. A benchmarking study reported such advantages. In their benchmark set, combining QPU sampling with classical postprocessing and decomposition can simultaneously achieve higher accuracy and shorter solving times than either pure quantum or pure classical approaches in more than 50 large and dense QUBO instances [10]. These empirical results support treating hybrid as an architectural principle rather than a simple concatenation of solvers.

For analytical convenience, existing hybrid schemes can be broadly organized into two paradigms. The first is sampling or sub-QUBO centered hybridization. In this paradigm, the original problem is partitioned into subproblems that fit within the programmable scale of a QPU, each subproblem is solved by annealing, and the returned samples are reintegrated into a global solution estimate. A representative implementation is the large neighborhood local search (LNLS), which in each iteration selects a random neighborhood subproblem, submits it to the QPU, and greedily updates the global state with the lowest-energy sample [52]. Raymond et al. demonstrated the effectiveness of this approach on lattice-structured spin-glass problems at twice the directly programmable scale of an Advantage QPU, and showed that lattice-specific preoptimized embeddings significantly outperform generic clique embeddings [52]. The earlier qbsolv utility and the subsequent dwave-hybrid framework

follow a similar decompose-and-recombine logic, where an outer classical loop manages subproblem generation and solution stitching while an inner quantum solver performs energy minimization over restricted neighborhoods [50]. Yarkoni et al. further observed that sub-QUBO centered (problem partitioning) is the most common approach among the real-world hybrid applications reviewed, because the number of available qubits remains small relative to practical problem scales [50]. Under this paradigm, quantum annealing essentially serves as a constrained subproblem solver whose sampling quality is jointly determined by embedding efficiency, chain length, and annealing parameters.

The second paradigm is orchestration-centered hybridization. Here, a classical exact solver retains control over global optimality certificates, while quantum or physics-inspired solvers are invoked as external oracles at specific stages of the search process. Peng et al. explored several integration points for Ising solvers within a branch-and-bound framework, including incumbent injection at the root node and node-level heuristic callbacks at interior nodes [19]. They evaluated their method on the QUBOLib benchmark set, and their best hybrid configurations reported up to eleven percent runtime reduction and seventeen percent node reduction relative to default Gurobi. However, they also noted that standalone node-level heuristic callbacks are computationally expensive and often counterproductive [19]. This observation indicates that in orchestration-centered schemes, deciding when and how to invoke the quantum component requires careful design rather than yielding benefits by default. Unlike the sampling-centered paradigm, orchestration-centered schemes preserve global optimality certificates, and the role of the quantum component is to accelerate the search rather than to replace the search framework itself. More generally, decomposition strategies that separate a combinatorial master problem from a linearly structured subproblem and iterate between them can also be placed under this paradigm, where the master problem is converted into a QUBO for quantum solving and the subproblem is handled by a classical solver.

Both paradigms face limitations in transparency and adaptability, though the specific manifestations differ. On the sampling-centered side, managed hybrid solvers available through the D-Wave Leap platform expose limited control over internal decomposition and parameterization; users typically submit a problem and receive a result. In contrast, the open-source dwave-hybrid framework allows customization of the workflow-level, including user-defined subproblem selection strategies, subsolver substitution, and adjustment of annealing parameters [50, 52]. However, Raymond et

al. showed that hybrid workflow performance is highly sensitive to annealing time, number of reads, embedding scheme and workflow variants, and that the optimal parameterization changes with both the time scale and problem structure [52]. This means that even within an open framework, designing a hybrid workflow that adapts across different problems and execution stages remains an unsolved engineering problem. On the orchestration-centered side, the effectiveness of the quantum component depends heavily on the match between subproblem scale and hardware capacity. Peng et al. observed that performance degrades significantly when the subsolver scale is artificially reduced [19]. These observations collectively point to a need that has not been adequately addressed: how to design transparent and adaptive hybrid orchestration mechanisms in which the division of labor between classical and quantum components can be dynamically adjusted according to problem characteristics and solving progress.

Because every application chapter in this thesis requires a systematic comparison between hybrid schemes and classical baselines, it is necessary to establish shared benchmarking principles here. The most fundamental element is the choice of baseline solvers. In combinatorial optimization, integer programming solvers such as Gurobi typically serve as exact optimality references, while simulated annealing and tabu search serve as heuristic references [10, 19]. For solution quality measurement, Kim et al. used a relative accuracy metric in their negative-energy benchmarking setting, defined as the ratio of a given solver's objective value to the best objective value found across all solvers [10]. This metric is intuitive when all solvers produce objective values of a consistent sign and the optimum corresponds to the minimum value. In settings where the objective takes positive values, crosses zero, or follows a different sign convention, a normalized optimality gap may serve as a more robust alternative. Regarding time measurement, Kim et al. reported QPU access time by combining programming time and sampling time while excluding problem reading time, queue waiting time, and network communication time [10]. Raymond et al. further noted that the full end-to-end wall-clock time of a distributed hybrid computation is difficult to precisely measure and may be significantly longer than the QPU access time alone [52]. Accordingly, QPU access time is an appropriate metric for solver benchmarking, but it is not sufficient for deployment-level latency evaluation.

In summary, hybrid quantum-classical architectures provide practical paths to overcome the scale bottleneck of current quantum hardware at both the sampling-driven and orchestration-driven levels. However, existing schemes still exhibit gaps in

formulation automation, transparent orchestration, and cross-domain methodological transferability. The next section systematically synthesizes these open issues from the literature and maps them to the four research gaps identified in Chapter 1.

2.6 Synthesis of literature and thesis gaps

The preceding five sections have established the methodological chain from classical modeling to hybrid execution: the shared structural characteristics of combinatorial decision optimization in networked systems (Section 2.1), classical mathematical programming foundations (Section 2.2), the QUBO intermediate abstraction layer (Section 2.3), quantum annealing and hardware-aware execution (Section 2.4), and hybrid quantum-classical optimization architecture (Section 2.5). The classical modeling and uncertainty treatment covered in Sections 2.1 and 2.2 draw on a mature body of literature. MILP [38], two-stage stochastic integer programming [39], and robust optimization [40] provide well-established source model foundations for the QUBO transformations that follow. Section 2.3 established the role of QUBO as an intermediate abstraction layer connecting classical formulations with downstream annealing-oriented and hybrid execution workflows [11, 12, 47]. However, QUBO modeling in existing literature remains heavily dependent on manual expert knowledge. Each step from constraint type identification to penalty coefficient setting and variable encoding precision selection lacks systematic and automated methodological support [12, 50]. Section 2.4 showed that hardware topology, embedding overhead, and sampling behavior collectively shape the practical solving capability of quantum annealing [53,54], but the vast majority of existing workflows assume a single static submission for a fixed problem instance and lack adaptation mechanisms for time-varying environments and multi-stage decisions [16]. Section 2.5 demonstrated progress in both sampling-driven and orchestration-driven hybrid paradigms for overcoming hardware scale bottlenecks [19,52], but sampling-driven schemes do not natively support mixed discrete-continuous decision making, while orchestration-driven schemes remain limited in formulation automation and transparent coordination [19, 50, 52]. Table 2.1 systematically maps these open issues to the four research gaps identified in Chapter 1, and indicates where each gap is addressed in subsequent chapters. The following four chapters respond to these gaps in the methodological progression summarized in Table 1.1.

Table 2.1: Mapping of literature foundations to research gaps and thesis chapters.

Research gap	Literature foundation in Chapter 2	Identified limitation
G1: Formulation dependence (Chapters 3 and 6)	QUBO reformulation relies on penalty embedding, slack introduction, and binary encoding of non-binary quantities (Sec. 2.3) [11, 12, 47]. Constrained optimization to QUBO conversion is well understood in principle [11], but requires manual, problem-specific effort for each new application [50]. Classical source models including MILP and two-stage stochastic integer programming (Sec. 2.2) [38, 39] provide the upstream formulations from which QUBO must be derived.	No systematic method exists for automating the full pipeline from a structured MILP to a hardware-aware QUBO with appropriate penalty weights and encoding precision. Existing tools such as PyQUBO provide syntax-level support but do not automate constraint analysis or penalty calibration [47, 50].
G2: Workflow rigidity (Chapter 4)	Quantum annealing executes a fixed problem Hamiltonian per submission (Sec. 2.4) [10, 53]. Hardware topology constrains embeddable problem size and structure [54]. Benchmarking confirms that solution quality depends on annealing parameters, embedding, and number of reads [10, 52].	Existing QA workflows treat each submission as an independent static instance. There is no established mechanism for adapting the formulation or execution strategy across sequential decision epochs in time-varying networked environments [16, 50].
G3: Limited decision expressiveness (Chapter 5)	QUBO natively encodes only binary decision variables (Sec. 2.3) [11, 47]. Continuous and integer variables require binary expansion, which increases problem size and amplifies noise sensitivity (Sec. 2.4) [10, 50]. Hybrid schemes reviewed in Sec. 2.5 partition problems at the QUBO level but do not coordinate discrete and continuous decisions within a single loop [19, 52].	Sampling-driven hybrid methods solve binary subproblems only. No reviewed scheme jointly optimizes discrete scheduling priorities and continuous resource allocation within one integrated decision architecture.
G4: Scalability bottleneck (Chapter 6)	Embedding overhead scales unfavorably with problem density and size (Sec. 2.4) [10, 54]. Hybrid LNLS and branch-and-bound methods partially address this through decomposition (Sec. 2.5) [19, 52], but managed hybrid solvers expose limited transparency [50] and parameter sensitivity remains high [52].	No reviewed framework combines automated QUBO formulation with decomposition-based hybrid orchestration in a transparent and scalable manner. The formulation step and the solving step remain disjoint in existing pipelines [16, 19].

Chapter 3

Stochastic Resource Planning for Metaverse Data Marketplaces

This chapter investigates the stochastic resource planning problem in metaverse data marketplaces. It corresponds to Stage 1 in Table 1.1, namely domain-grounded formulation and empirical validation. Chapter 2 positions QUBO as an intermediate abstraction layer connecting application-level modeling to downstream solution execution. This chapter verifies the practical operability of that positioning through a complete end-to-end instance. The research scenario involves optimizing edge device subscriptions for metaverse service providers under data sovereignty constraints. The chapter constructs a two-stage stochastic integer programming model, transforms it into a QUBO representation, and validates the formulation on a 550-qubit coherent Ising machine. This pipeline also addresses the research Gap G1 by revealing the specific manifestations of formulation dependence in existing QUBO workflows. The content of this chapter is based on [1] and has been extended in experimental analysis and methodological discussion.

3.1 Introduction

Constructing high-fidelity transportation digital twins requires large volumes of real-time data collected from specific geographic regions. However, data sovereignty regulations restrict cross-border data acquisition, particularly affecting autonomous driving companies that rely on localized training data. A metaverse service provider (MSP) acts as a data intermediary. It subscribes to raw sensing data from edge

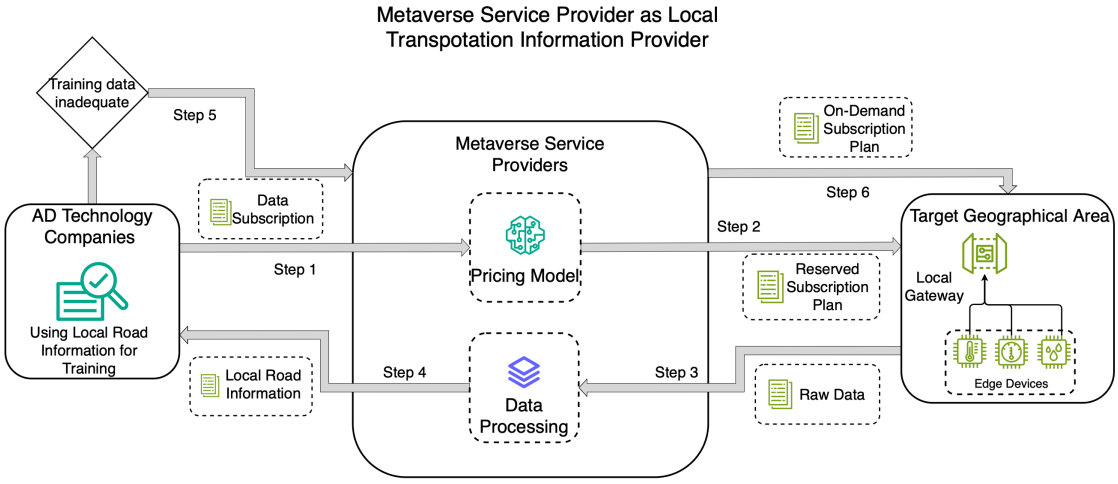


Figure 3.1: The workflow of Metaverse Service Provider with two-phase training data.

devices deployed in a target geographical area (TGA) and delivers processed transportation information to its clients. The edge devices are not directly owned by the MSP. Instead, their data resources are accessed through subscription plans. The central optimization problem facing the MSP is how to allocate resources optimally between reserved and on-demand subscriptions under uncertain demand, so as to minimize total operational cost [55]. Reserved plans offer lower unit costs, but oversubscription leads to wasted resources. On-demand plans provide flexibility at a higher unit price. This tradeoff between reservation and on-demand procurement forms a canonical two-stage stochastic decision structure.

Figure 3.1 illustrates the end-to-end workflow of the MSP. An autonomous driving technology company submits a service request to the MSP (Step 1). The pricing model of the MSP then generates a reserved subscription plan and sends it to the target geographical area (Step 2). Edge devices within the region collect raw data and transmit them back to the MSP (Step 3). The MSP processes these data into structured local transportation information and delivers the result to the client (Step 4). If the training data are insufficient, the client issues a supplemental request (Step 5), and the pricing model generates an on-demand subscription plan accordingly (Step 6). Steps 3 through 4 are then repeated to close the data loop. This iterative feedback mechanism allows the MSP to handle both the deterministic reserved decision and the stochastic on-demand replenishment decision within a single optimization cycle.

To model this decision structure, the chapter constructs a two-stage stochastic in-

teger programming (SIP) formulation. The first stage determines the subscription status and the number of reserved data bundles for each edge device. The second stage, executed after demand scenarios are realized, decides the number of on-demand bundles to purchase in order to cover any remaining deficit. The objective function of the SIP model consists of a deterministic reservation cost component and a probability-weighted on-demand cost component. To enable execution on quantum computing backends, the chapter transforms the SIP model into a QUBO representation. The transformation involves binary encoding of integer variables and penalty-based embedding of constraints. During encoding, an adaptive bit-width truncation strategy is adopted, limiting the number of encoding bits to $K = \min(\lceil \log_2(X+1) \rceil, 5)$ in order to control the dimensional growth of the QUBO matrix. The resulting QUBO instances are solved and compared on a 550-qubit coherent Ising machine (CIM), the Gurobi exact solver, and the simulated annealing solver dwave-neal.

The specific contributions of this chapter are as follows. First, it designs an end-to-end MSP-centric workflow that unifies reserved and on-demand subscription decisions under data sovereignty constraints within a two-stage stochastic optimization framework. Second, it introduces an adaptive bit-width truncation and low-coupling parameter design strategy that effectively controls the number of binary variables during the SIP-to-QUBO transformation, keeping the QUBO matrix dimension at the order of $O(|\mathcal{W}| \times |\mathcal{E}| \times (K + |\Omega| \times K))$. Third, it completes the actual solving validation of this QUBO model on a 550-qubit CIM. Experimental results show that the CIM obtains high-quality solutions within millisecond-level computation times, significantly outperforming simulated annealing in speed while approaching the solution quality of the Gurobi exact solver. These results provide the first complete empirical validation of the classical-modeling-to-QUBO-to-hardware-execution methodological path proposed in Chapter 2.

The remainder of this chapter is organized as follows. Section 3.2 presents the complete two-stage stochastic integer programming formulation. Section 3.3 details the transformation from SIP to QUBO. Section 3.4 conducts the experimental evaluation and analyzes the results. Section 3.5 summarizes the chapter.

3.2 System Model

The workflow described in Section 3.1 involves two sequential decision points, which are formalized below as a two-stage stochastic program. This section formulates the

Table 3.1: Parameters and Decision Variables for Resource Subscription Model

System Model Parameters	
Notation	**Description**
$\mathcal{W}$	Set of Metaverse Service Providers
Ω	Set of possible scenarios
$\mathcal{E}$	Set of edge devices
X	Maximum reserved bundles per subscription
$C_e^{(r,\text{memb})}$	Membership cost per edge
$C_e^{(r,\text{trans})}$	Reserved data bundle transmission cost per edge
$C_e^{(o,\text{trans})}$	On-demand data bundle transmission cost per edge
Scenario Parameters	
Notation	**Description**
λ_s	Scenario index
$P(\lambda_s)$	Probability of scenario λ_s
$S_{w,e}(\lambda_s)$	Similarity score in scenario λ_s
$\bar{F}_w(\lambda_s)$	Actual demand of MSP w in λ_s
Decision Variables	
Notation	**Description**
$m_{w,e}^{(r)}$	Binary subscription indicator
$\tilde{m}_{w,e}^{(r)}$	Reserved bundles purchased
$m_{w,e}^{(o)}(\lambda_s)$	On-demand bundles in scenario λ_s

cost minimization problem addressing both reservation and on-demand subscription scenarios, with the objective function explicitly handling uncertainty.

The objective function consists of costs from two stages: a deterministic reservation phase and a stochastic on-demand phase. The first part of the objective, $\sum_{w\in\mathcal{W}}\sum_{e\in\mathcal{E}}\left(m_{w,e}^{(r)}C_e^{(r,\text{memb})}+\tilde{m}_{w,e}^{(r)}C_e^{(r,\text{trans})}\right)$, represents the first-stage reservation cost. In this stage, MSPs purchase memberships for specific edge devices and pay for a reserved number of data bundle transmissions. Notably, the data bundle encompasses the entire process of data collection, semantic extraction, and data transformation, with the core semantic extraction carried out by the edge device providers.

$$\min_{m_{w,e}^{(r)}, \tilde{m}_{w,e}^{(r)}, m_{w,e}^{(o)}(\lambda_s)} :$$

$$\sum_{w \in \mathcal{W}} \sum_{e \in \mathcal{E}} \left(m_{w,e}^{(r)} C_e^{(r,\text{memb})} + \tilde{m}_{w,e}^{(r)} C_e^{(r,\text{trans})} \right) + \mathbb{E}\left[Q\left(m_{w,e}^{(o)}(\lambda_s) \right) \right],$$

where

$$Q\left(m_{w,e}^{(o)}(\lambda_s) \right) = \sum_{\lambda_s \in \Omega} P(\lambda_s) \times \sum_{w \in \mathcal{W}} \sum_{e \in \mathcal{E}} m_{w,e}^{(o)}(\lambda_s) C_e^{(o,\text{trans})}$$

The second stage addresses the uncertainty inherent in the demand for real-world data. Following the two-stage SIP framework introduced in Section 2.2, each scenario $\lambda_s \in \Omega$ represents a distinct possible future state of the transportation environment, and is assigned a probability $P(\lambda_s)$ estimated from historical traffic and meteorological data [55]. The parameters dependent on these scenarios, such as the total data demand $\bar{F}_w(\lambda_s)$ and the data similarity score $S_{w,e}(\lambda_s)$, are defined for each state accordingly.

This leads to the second part of the objective function, which calculates the expected on-demand cost across all scenarios: $\sum_{\lambda_s \in \Omega} P(\lambda_s) \times \sum_{w \in \mathcal{W}} \sum_{e \in \mathcal{E}} m_{w,e}^{(o)}(\lambda_s) C_e^{(o,\text{trans})}$. Here, $m_{w,e}^{(o)}(\lambda_s)$ is the number of on-demand data bundles purchased in a specific scenario λ_s, and this cost is weighted by the probability of the scenario $P(\lambda_s)$, ensuring that the total objective minimizes the expected cost under uncertainty.

The SIP model is subject to the following constraints.

$$\mathcal{C}1 : \tilde{m}_{w,e}^{(r)} \leq m_{w,e}^{(r)} X, \quad \forall w \in \mathcal{W}, \forall e \in \mathcal{E}$$

The constraint $\mathcal{C}1$ requires that the subscription of an edge device be made before getting any data bundle from that device in the reserved price level, i.e. $m_{w,e}^{(r)} = 1$. Additionally, even after subscribing, the MSP can purchase at most X reserved data bundles from that edge device.

$$\mathcal{C}2 : \sum_{e \in \mathcal{E}} \tilde{m}_{w,e}^{(r)} S_{w,e}(\lambda_s) + \sum_{e \in \mathcal{E}} m_{w,e}^{(o)}(\lambda_s) \geq \bar{F}_w(\lambda_s), \quad \forall w \in \mathcal{W}, \forall \lambda_s \in \Omega$$

Constraint $\mathcal{C}2$ ensures that, in each scenario λ_s, the total amount of data from both reserved and on-demand purchases meets or exceeds the demand $\bar{F}_w(\lambda_s)$. The term $\tilde{m}_{w,e}^{(r)} S_{w,e}(\lambda_s)$ represents the effective number of data bundles, adjusted by the similarity score $S_{w,e}(\lambda_s)$, which reflects how well the data from edge device e matches the

MSP requirements in scenario λ_s.

The subscription indicator $m_{w,e}^{(r)} \in \{0,1\}$ is binary, and both the reserved bundle count $\tilde{m}_{w,e}^{(r)} \in \mathbb{Z}^+$ and the on-demand bundle count $m_{w,e}^{(o)}(\lambda_s) \in \mathbb{Z}^+$ are non-negative integers.

3.3 QUBO Transformation

This section transforms the SIP model developed in Section 3.2 into an equivalent QUBO formulation. As introduced in Section 2.3, QUBO provides a shared binary abstraction that connects constrained optimization models to quantum annealing and quantum-inspired solution backends. The transformation involves two steps: binary encoding of integer decision variables and penalty-based embedding of inequality constraints. To control the dimensional growth of the resulting QUBO matrix, this chapter adopts an adaptive bit-width truncation strategy combined with a low-coupling parameter design, keeping the total number of binary variables at the order of 10^3.

3.3.1 Binary Encoding of Integer Variables

The SIP model in Section 3.2 contains integer decision variables $\tilde{m}_{w,e}^{(r)}$ and $m_{w,e}^{(o)}(\lambda_s)$ that must be expressed in binary form before QUBO construction. Each integer variable is encoded through K-bit binary expansion with $K = \min(\lceil \log_2(X+1) \rceil, 5)$, following the adaptive truncation strategy described in Section 3.1. The core encoding relationships are defined as follows:

$$\tilde{m}_{w,e}^{(r)} = \sum_{k=1}^{K} 2^{k-1} b_{w,e,k}^{(r)}, \quad m_{w,e}^{(o)}(\lambda_s) = \sum_{l=1}^{L} 2^{l-1} b_{w,e,l}^{(o)}(\lambda_s) \tag{3.1}$$

where $b_{w,e,k}^{(r)}, b_{w,e,l}^{(o)} \in \{0,1\}$ are binary decision variables. When $K = 5$, the quantization error remains below 4%. This encoding reduces the variable count from $O(|\mathcal{W}| \times |\mathcal{E}| \times |\Omega|)$ integer variables to $O(|\mathcal{W}| \times |\mathcal{E}| \times (K + |\Omega| \times K))$ binary variables.

3.3.2 Penalty-Based Constraint Embedding

Following the general penalization procedure introduced in Section 2.3, the complete QUBO objective combines the original cost terms with quadratic penalty terms for

constraint enforcement:

$$H = \underbrace{\sum_{w,e} \left[C_e^{(r,\text{memb})} m_{w,e}^{(r)} + C_e^{(r,\text{trans})} \tilde{m}_{w,e}^{(r)} \right]}_{\text{Membership costs}} + \underbrace{\mathbb{E}_\lambda \left[\sum_{w,e} C_e^{(o,\text{trans})} m_{w,e}^{(o)}(\lambda) \right]}_{\text{On-demand costs}} + \underbrace{\alpha P_{C_1} + \beta P_{C_2}}_{\text{Constraints}} \tag{3.2}$$

The constraint terms implement the problem's physical requirements through quadratic penalties. Since both $\mathcal{C}1$ and $\mathcal{C}2$ are inequality constraints, their conversion to QUBO form requires slack variables that transform each inequality into an equality before squaring. This follows the standard procedure reviewed in Section 2.3.

For C_1, the original per-pair inequality $\tilde{m}_{w,e}^{(r)} \leq m_{w,e}^{(r)} \cdot X$ is converted by adding a non-negative integer slack variable $s_{w,e}^{(1)}$ to absorb the gap between the right-hand side and the left-hand side:

$$P_{C_1} = \sum_{w \in W} \sum_{e \in E} \left(\tilde{m}_{w,e}^{(r)} + s_{w,e}^{(1)} - m_{w,e}^{(r)} \cdot X \right)^2 \tag{3.3}$$

where $s_{w,e}^{(1)}$ is a non-negative integer slack variable encoded using binary expansion:

$$s_{w,e}^{(1)} = \sum_{k=1}^{K_s} 2^{k-1} \, b_{w,e,k}^{(s_1)}, \quad b_{w,e,k}^{(s_1)} \in \{0, 1\} \tag{3.4}$$

The upper bound of the slack equals X (attained when $m_{w,e}^{(r)} = 1$ and $\tilde{m}_{w,e}^{(r)} = 0$), so the required number of slack bits is $K_s = \lceil \log_2(X + 1) \rceil$.

For C_2, the original inequality requires that the total supply meets or exceeds the demand, only the undersupply is infeasible. Therefore, a slack variable $s_{w,\lambda_s}^{(2)}$ is introduced to represent the permissible oversupply:

$$P_{C_2} = \sum_{w \in W} \sum_{\lambda_s \in \Omega} \left(\sum_{e \in E} \left[S_{w,e}(\lambda_s) \, \tilde{m}_{w,e}^{(r)} + m_{w,e}^{(o)}(\lambda_s) \right] - \bar{F}_w(\lambda_s) - s_{w,\lambda_s}^{(2)} \right)^2 \tag{3.5}$$

where $s^{(2)}_{w,\lambda_s}$ is encoded as:

$$s^{(2)}_{w,\lambda_s} = \sum_{l=1}^{L_s} 2^{l-1}\, b^{(s_2)}_{w,\lambda_s,l}, \quad b^{(s_2)}_{w,\lambda_s,l} \in \{0,1\} \tag{3.6}$$

The upper bound of this slack is determined by the maximum possible oversupply, and the required number of bits is $L_s = \lceil \log_2(s^{(2)}_{\max} + 1) \rceil$. The penalty coefficients α and β must be large enough to make constraint violation energetically unfavorable, as discussed in Section 2.3.

Through empirical tuning, we selected $\alpha = 10000$ and $\beta = 100$. This hierarchical setting, where the penalty for the fundamental subscription logic (C1) is much greater than for the service demand (C2), ensures that all solutions returned from the CIM were feasible while also preserving a reasonable structure for the quantum energy landscape. Although these values proved to be effective, a detailed sensitivity analysis of these parameters is a potential avenue for future work.

The final model contains $O(|\mathcal{W}||\mathcal{E}|(K + |\Omega|L))$ binary variables, including both decision and slack variables.

3.4 Experimental Evaluation

This section evaluates the QUBO formulation developed in Section 3.3 on three computational backends: a 550-qubit coherent Ising machine (CIM), the Gurobi exact solver in QUBO mode, and the dwave-neal simulated annealing sampler. The evaluation examines two dimensions: solution quality and computation time across three problem scales.

We first formulated the problem as an SIP model and solved it with the Gurobi optimizer as a baseline. After converting the SIP into a QUBO formulation, we solved the same QUBO matrix with three solvers: the `solve_qubo` method from `gurobi_optimods` (classical exact solver), the `SimulatedAnnealingSampler` from `dwave-neal` [26]), and the QBoson CPQC-550 CIM (quantum hardware) [24]. The QUBO matrices were constructed using PyQUBO [56] and the Kaiwu SDK as modeling tools, respectively.

The QUBO model was submitted to the QBoson CPQC-550, a 550-qubit coherent photonic quantum computer, through the Kaiwu SDK. The SDK maps the QUBO matrix onto the quantum processing unit and initiates the physical annealing process.

All CIM results reported below are from direct execution on quantum hardware.

Three distinct problem scales (small, medium, and large) were simulated with systematic parameter configurations as summarized in Table 3.2. The potential scenario range was fixed between 2 and 5, while the quantity of edge devices ranged from 5 to 25. The total demand for data bundles followed a Poisson distribution. The cost ratios of subscription fee, reserved transmission and on-demand purchase were set at 10: 1: 5 based on empirical market data [57].

Table 3.2: Parameter Configurations Across Problem Scales

Parameter	**Small (S)**	**Medium (M)**	**Large (L)**
Number of MSPs	1	2	5
Edge Devices	5	10	25
Service Scenarios	2	3	5
Data Bundles Demand	2,000	6,000	15,000

3.4.1 Solution Quality and Computation Time

Fig. 3.2 compares the solution quality of these three approaches on different problem scales. The results highlight a trade-off between computational time and solution optimality. As shown in Fig. 3.2, the classical Gurobi optimizer consistently found the lowest objective values, indicating the highest quality solutions across all problem scales. However, the CIM produced highly competitive solutions, achieving objective values that were only marginally higher than those of Gurobi's, while both significantly outperformed the `dwave-neal` simulated annealing solver. As shown in Fig. 3.3 and Table 3.3, the CIM maintains a millisecond-scale computation time across all problem sizes, with execution times of 0.001718s (S-scale), 0.001687s (M-scale), and 0.005235s (L-scale). The slight time reduction from small to medium scale is likely attributable to system overhead and measurement fluctuations in such short execution windows. Across these three scales, the CIM execution time does not exhibit exponential growth. However, this observation is based on a limited number of problem sizes and does not constitute a rigorous characterization of asymptotic complexity.

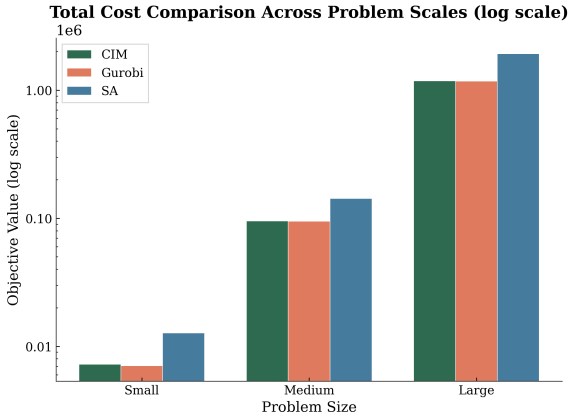


Figure 3.2: Comparison of logarithmic total costs obtained via CIM (quantum hardware), Gurobi (QUBO mode), and `dwave-neal` simulated annealing under small, medium, and large problem configurations defined in Table 3.2

Table 3.3: Execution Time Comparison Across Problem Scales

Method	**S-scale (s)**	**M-scale (s)**	**L-scale (s)**
CIM	0.001718	0.001687	0.005235
Gurobi (QUBO)	0.0311	0.0674	1.5235
SA (dwave-neal)	1.42	13.78	310.765579

3.4.2 Analysis of Encoding and Scalability

The experimental results allow further analysis of the encoding strategy and its impact on scalability. By introducing binary encoding strategies that map multidimensional decision variables ($m_{w,e}^{(r)}$, $\tilde{m}_{w,e}^{(r)}$, $m_{w,e}^{(o)}(\lambda_s)$) to compact QUBO representations, reduced problem dimensionality. Specifically, the bit decomposition technique implemented in the code, which restricts $K = min(ceil(log_2(X+1)), 5)$, significantly reduces the size of the QUBO matrix.

These results provide evidence for the practical feasibility of CIM-based optimization in this problem setting. While classical solvers exhibit comparable optimality gaps ($< 5\%$) in small-scale scenarios, they suffer from prohibitive exponential time complexity growth as problem dimensions increase. Across the three tested problem scales, the execution time of the CIM did not show exponential growth with increasing problem size. Current limitations include the sensitivity of the CIM to penalty coefficient calibration and the absence of minor embedding overhead that is required on superconducting annealers with sparse connectivity. A more thorough investigation of penalty sensitivity is left to future work.

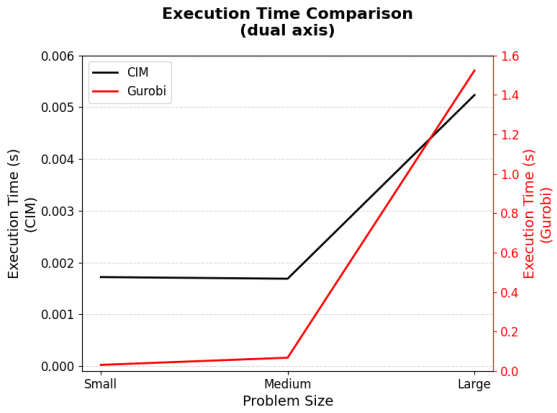


Figure 3.3: Comparison of computation time between the coherent Ising machine, left axis, and Gurobi, right axis, for solving the target QUBO formulation across small, medium, and large problem scales.

3.5 Summary

This chapter formulated the stochastic resource planning problem in metaverse data marketplaces as a two-stage SIP model and transformed it into a QUBO representation through binary encoding and penalty-based constraint embedding. The resulting formulation was validated on a 550-qubit CIM, the Gurobi exact solver, and the dwave-neal simulated annealing sampler. Experimental results show that the CIM achieves millisecond-level computation times with solution quality within 5% of the Gurobi optimum, while dwave-neal exhibits significantly longer runtimes without quality advantage. These results confirm the practical operability of the classical-to-QUBO-to-hardware pipeline proposed in Chapter 2. At the same time, the formulation process in this chapter relied entirely on manual variable encoding and penalty calibration, illustrating the formulation dependence identified as Research Gap G1. The next chapter extends the methodology from static single-instance optimization to dynamic adaptation under time-varying network conditions.

Chapter 4

Two-Stage Resilient Topology Control for UAV Communication Networks

4.1 Introduction

This chapter investigates the resilient topology control problem in dynamic UAV communication networks. It corresponds to Stage 2 in Table 1.1, namely two-stage recourse-aware dynamic adaptation. Chapter 3 validates the end-to-end conversion pipeline from classical planning models to QUBO in a metaverse data marketplace scenario, but that pipeline completes after a single submission and does not adapt to time-varying conditions. This chapter builds on that foundation by introducing a mechanism that separates offline candidate generation from online lightweight selection, thereby overcoming the limitation of fixed single-instance workflows in dynamic environments. This design directly addresses Research Gap G2, namely the absence of a native recourse mechanism in existing QUBO-based optimization workflows. The content of this chapter is based on [3] and has been extended in experimental analysis and methodological discussion.

Effective coordination of multi-UAV swarms is critical for missions in emergency response, logistics, and infrastructure inspection [58]. However, maintaining stable and efficient communication links presents a significant challenge. The high mobility of UAVs and unpredictable environmental factors such as wind gusts, node drift, and temporary line-of-sight obstruction cause frequent changes in the network topology.

In such dynamic scenarios, a single optimized topology is often fragile. Even a small variation in node position or channel condition can lead to a sharp degradation in network performance [59]. A more resilient strategy is to pre-calculate a set of high-quality and structurally diverse candidate topologies. This approach allows UAVs to rapidly switch to the most suitable configuration with minimal computational cost, enabling continuous adaptation to real-time conditions. However, generating this diverse topology set requires repeatedly solving the topology optimization problem, which is computationally intensive. Conventional approaches, including exact algorithms, heuristic methods, and deep reinforcement learning, face fundamental limitations in meeting the stringent real-time and onboard resource constraints of dynamic UAV networks [60, 61].

As discussed in the third paragraph of Section 2.4, the sampling behavior of QA differs structurally from that of classical stochastic solvers and can produce more diverse sets of near-optimal solutions within the same time budget. This diversity property is central to the present chapter, where the goal is to generate a set of candidate topologies that are not only individually high-quality but also structurally complementary to one another.

Building on the diversity sampling capability of QA described above, this chapter proposes a two-stage framework that decouples computationally intensive topology optimization from online decision making. In the offline phase, the UAV topology control problem is formulated as a QUBO model. As discussed in Chapter 2, QUBO serves as the native input format for QA, allowing the optimization problem to be directly mapped onto quantum hardware for parallel exploration of complex solution landscapes [62]. The QUBO model incorporates penalty terms to ensure load balancing, reduce node overload risk, and avoid single-point failures. In dynamic UAV networks, such structural imbalances can amplify rapidly as the topology changes, leading to severe communication degradation [63]. To further enhance structural differentiation among candidate topologies, QA is employed with an iterative similarity penalty mechanism to generate a structurally diverse and complementary set of candidates. During deployment, a lightweight classical evaluation mechanism rapidly selects the most suitable topology based on real-time link stability and Signal-to-Interference-plus-Noise Ratio (SINR), enabling fast adaptation while maintaining network quality.

Figure 4.1 illustrates the two stages of the proposed framework. The left panel depicts a fragile UAV swarm, where a centralized topology may experience a complete

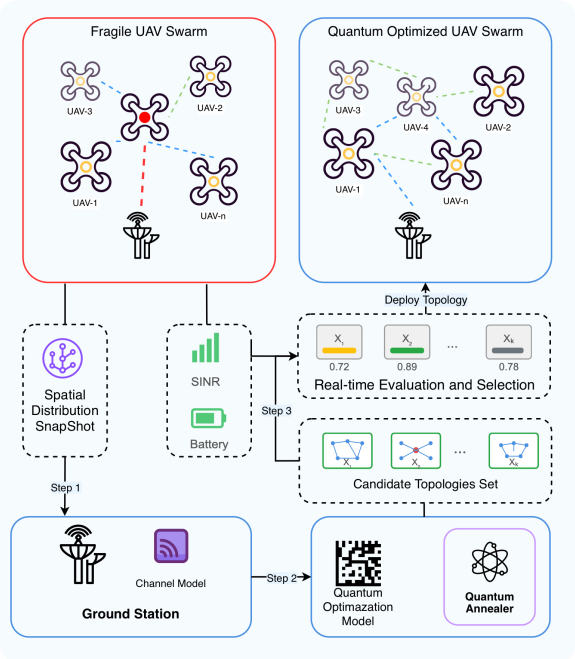


Figure 4.1: Two-stage topology control framework: offline generation of diverse candidate topologies via QA (Steps 1-2), and online lightweight selection based on real-time network conditions (Step 3).

communication breakdown if a single critical node fails. The right panel shows a more balanced and decentralized structure produced by QA. Steps 1 and 2 in the figure correspond to the offline stage, where the ground station constructs a QUBO model based on the spatial snapshot and channel parameters, then executes QA to produce a set of candidate topologies. Step 3 represents the online stage, in which the UAVs continuously evaluate these candidates using real-time SINR and remaining energy feedback, and select the most suitable topology for deployment, enabling fast and efficient reconfiguration.

The main contributions of this chapter are summarized as follows. First, a two-stage quantum-assisted topology control framework is proposed that decouples heavy offline optimization from lightweight online decision making, enabling scalable and adaptive topology control in dynamic UAV networks. This framework is compatible with Software-Defined Networking (SDN) and Open Radio Access Network (O-RAN) architectures, supporting practical deployment. Second, the offline stage formulates the topology control task as a QUBO problem solved via QA, which exploits quantum parallelism to explore multiple topologies simultaneously and generate a structurally diverse set of high-quality candidates. Third, the online stage employs a lightweight

evaluation mechanism that selects the most suitable topology based on real-time SINR and residual energy, achieving fast adaptation with minimal onboard computation. Fourth, experimental results show that the proposed framework improves performance retention by 6.6% over a static topology baseline, while QA achieves a 5.15% reduction in objective value and a 28.3% increase in solution diversity compared to simulated annealing.

The remainder of this chapter is organized as follows. Section 4.2 reviews related work on UAV network topology control. Section 4.3 presents the system model and QUBO formulation. Section 4.4 reports on the experimental setup and key results. Section 4.5 concludes the chapter.

4.2 Related Work

Much of the existing literature on UAV network topology control relies on conventional optimization methods. Many studies formulate these problems as mixed-integer linear programs (MILP) to find globally optimal solutions for tasks such as node placement and trajectory planning [60]. However, as discussed in Chapter 2, these models are NP-hard, and their computational complexity makes them impractical for large and highly dynamic networks. To avoid high computational costs, heuristic and metaheuristic algorithms such as Particle Swarm Optimization and Genetic Algorithms are often applied to tasks like path planning and formation reconfiguration [64]. While often faster, these methods lack formal convergence guarantees and have unpredictable computation times, making them unreliable for real-time applications with strict deadlines [61].

More recently, deep reinforcement learning has been applied to UAV topology optimization [64], but the substantial computational and power requirements of deep neural networks conflict with the strict constraints of UAV platforms, reducing mission endurance. Quantum computing, particularly QA, offers a new approach to overcome these classical computational bottlenecks. QA has been successfully applied to several NP-hard problems in networking, such as solving scheduling problems in wireless networks to maximize throughput [65]. However, a central challenge remains. Conventional methods are too slow, unreliable, or resource-intensive for dynamic UAV environments [66], while QA applications have largely focused on static network problems, and hardware limitations make QA unsuitable for direct real-time control.

In summary, existing approaches either focus on static optimization or rely on heuristic and learning-based methods whose computation time is unpredictable, making them unsuitable for time-critical UAV communication. This limitation aligns with Research Gap G2 identified in Chapter 1: existing QUBO-based workflows are organized around a fixed single submission and lack a native mechanism for continuous adaptation in time-varying environments. A framework is therefore needed that can efficiently generate high-quality candidate topologies offline and support rapid adaptation during flight within the operational limits of UAV hardware. The following sections present the system model and QUBO formulation designed to bridge this gap.

4.3 System Model and Proposed Framework

This section develops the mathematical formulation for the UAV topology control framework. The UAV communication network is first modeled as an undirected graph, and an optimization objective that jointly considers throughput maximization and structural fragility penalization is defined. The main notations used in this section are summarized in Table 4.1. The objective is then expressed in the standard QUBO form defined in Section 2.3, enabling direct execution on quantum annealing hardware. Finally, the section describes how the resulting candidate topologies are integrated into a real-time decision mechanism and discusses the compatibility of this framework with SDN/O-RAN architectures.

Table 4.1: List of Key Notations Used in the Proposed Framework.

Symbol	Description
$\mathcal{V}, N$	Set of UAVs and total number of nodes in the network
x_{ij}	Binary variable indicating whether the communication link between UAV i and UAV j is activated
$\mathbf{x}$	Decision vector representing one feasible UAV topology, namely a set of active links
C_{ij}	Channel capacity between UAV i and UAV j, computed from the Shannon capacity formula with an SNR gap approximation for practical modulation
$T(\mathbf{x})$	Total achievable throughput of topology $\mathbf{x}$
$F(\mathbf{x})$	Structural fragility penalty reflecting hub load imbalance
$\mathcal{N}(k)$	Neighbor set of node k
α, β	Weight parameters balancing throughput and robustness
$\mathbf{Q}$	QUBO matrix constructed from the objective coefficients at the ground control station
$\mathcal{C}$	Set of UAV topologies generated by the quantum annealer in the offline stage
$S_i(t)$	Real-time utility score of topology $\mathbf{X}_i$ at time t
$E_l(t)$	Remaining energy of node l at time t
$w_{\text{perf}}, w_{\text{life}}$	Online weighting parameters balancing throughput and network longevity

4.3.1 Network Model and Objective Function

We model the UAV communication network as an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of UAVs with $|\mathcal{V}| = N$, and $\mathcal{E}$ represents all potential communication links between UAVs. Each link $(i, j) \in \mathcal{E}$ is associated with a binary decision variable $x_{ij} \in \{0, 1\}$, indicating whether the link is activated. The set of all decision variables forms the decision vector $\mathbf{x}$, which defines one feasible topology.

The quality of a topology is measured by an objective function $C(\mathbf{x})$ that captures the trade-off between the overall throughput and structural robustness:

$$\min_{\mathbf{x}} C(\mathbf{x}) = -\alpha\, T(\mathbf{x}) + \beta\, F(\mathbf{x}), \tag{4.1}$$

where $T(\mathbf{x})$ represents the proxy for total network throughput and $F(\mathbf{x})$ penalizes topological fragility. The parameters α and β control the trade-off between these objectives. These weights are mission dependent and can be tuned according to application priorities. For example, missions requiring continuous data transmission such as UAV monitoring favor larger α, while long-endurance or patrol tasks prefer higher β to emphasize structural robustness. To enable real-time quantum optimization, we adopt the Aggregate Link Capacity (sum of all active link capacities) as a part of the objective function, defined as:

$$T(\mathbf{x}) = \sum_{(i,j)\in\mathcal{E}} C_{ij}\, x_{ij}. \tag{4.2}$$

Here C_{ij} denotes the surrogate link capacity between UAV i and j, computed as $C_{ij} = B \log_2\big(1 + \gamma_{ij}/\Gamma\big)$ with $\Gamma = 2$.

The fragility term represents the vulnerability of a topology to node overload and is formulated as follows.

$$F(\mathbf{x}) = \sum_{k\in\mathcal{V}} \left(\sum_{j\in\mathcal{N}(k)} C_{kj}\, x_{kj} \right)^2. \tag{4.3}$$

This term penalizes topologies where a small number of UAVs carry disproportionately high traffic loads. The quadratic form enforces balanced traffic distribution, leading to more decentralized and resilient network structures.

4.3.2 Stage 1: Offline Strategic Computation

The objective function can be reformulated into the standard QUBO form $\min_{\mathbf{x}} \mathbf{x}^T \mathbf{Q} \mathbf{x}$ defined in Section 2.3, where $\mathbf{Q}$ is a symmetric matrix whose elements are derived from the throughput and fragility coefficients. This mapping allows the problem to be directly processed by a quantum annealer.

Algorithm 1: Topology Set Generation with Frequency-Based Penalty

1: *Input:* Initial QUBO matrix $\mathbf{Q}_0$, number of rounds R, samples per round k
2: *Output:* Topology set $\mathcal{C}$
3: Initialize $\mathcal{C} \leftarrow \emptyset$, $\mathbf{Q} \leftarrow \mathbf{Q}_0$
4: **for** $r = 1$ to R **do**
5: $\quad \{\mathbf{X}_{r,1}, \ldots, \mathbf{X}_{r,k}\} \leftarrow \text{QuantumAnnealer}(\mathbf{Q}, k)$
6: $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{X}_{r,1}, \ldots, \mathbf{X}_{r,k}\}$
7: $\quad$ Compute frequency matrix $f_{ij} = \frac{1}{k} \sum_{m=1}^{k} x_{ij}^{(r,m)}$
8: $\quad \mathbf{P} \leftarrow \lambda \, \text{diag}(f_{ij})$
9: $\quad \mathbf{Q} \leftarrow \mathbf{Q} + \mathbf{P}$
10: **end for**
11: *return* $\mathcal{C}$

As summarized in **Algorithm 1**, the offline procedure iteratively updates the QUBO matrix to promote exploration and prevent redundant solutions. At the beginning of the process (lines 1–3), the ground control station initializes the QUBO matrix $\mathbf{Q}_0$, specifies the number of annealing rounds R, and the number of samples k to be drawn from each run.

For each round $r \in R$ (lines 4–9), the quantum annealer executes a single annealing process on the current QUBO matrix and outputs k candidate topologies $C = \{\mathbf{X}_{r,1}, \ldots, \mathbf{X}_{r,k}\}$, which are merged into the candidate set $\mathcal{C}$. To encourage structural diversity, a similarity penalty matrix $\mathbf{P}$ is constructed according to the occurrence frequency of active links across these k topologies. Specifically, for each link (i, j), its activation frequency is computed as

$$f_{ij} = \frac{1}{k} \sum_{m=1}^{k} x_{ij}^{(r,m)}, \tag{4.4}$$

and a penalty proportional to this frequency is applied to the corresponding diagonal term:

$$P_{ij,ij} \leftarrow P_{ij,ij} + \lambda \, f_{ij}. \tag{4.5}$$

The QUBO matrix is then updated as $\mathbf{Q} \leftarrow \mathbf{Q} + \mathbf{P}$ before the next round, guiding subsequent annealing runs toward unexplored configurations. This iterative quantum–penalty process continues until sufficient distinct topologies are generated, forming the candidate set $\mathcal{C}$ used in the online stage.

4.3.3 Stage 2: Real-time Evaluation and Selection

Algorithm 2: Real-time Utility Evaluation and Topology Switching

1: *Input:* Candidate topology set $\mathcal{C} = \{\mathbf{X}_1, \ldots, \mathbf{X}_k\}$, weights w_{perf}, w_{life}
2: *Output:* Selected topology for the current deployment $\mathbf{X}_{\text{sel}}$
3: Initialize $\mathbf{X}_{\text{current}}$
4: **repeat**
5: Collect real-time metrics: $\{\text{SINR}_{jk}(t)\}, \{E_l(t)\}$
6: **for all** $\mathbf{X}_i \in \mathcal{C}$ **do**
7: Compute $S_i(t)$ using Eq. (6)
8: **end for**
9: $i^* \leftarrow \arg\max_i S_i(t)$
10: $\mathbf{X}_{\text{sel}} \leftarrow \mathbf{X}_{i^*}$
11: **if** $\mathbf{X}_{\text{sel}} \neq \mathbf{X}_{\text{current}}$ **then**
12: $\mathbf{X}_{\text{current}} \leftarrow \mathbf{X}_{\text{sel}}$
13: BroadcastSwitchCommand($\mathbf{X}_{\text{current}}$)
14: **end if**
15: **until** termination condition

The second stage operates on UAVs during flight and is designed to be lightweight. The swarm receives the precomputed candidate set $C = \{\mathbf{X}_{r,1}, \ldots, \mathbf{X}_{r,k}\}$ from the offline stage and continuously evaluates these topologies under current network conditions. Because link quality and energy vary rapidly, the online stage focuses on real-time adaptation by incorporating instantaneous SINR and residual energy, whereas the offline stage emphasizes long-term communication stability. As summarized in **Algorithm 2**, the online process executes a periodic evaluation–selection loop that determines the most suitable topology at each decision interval.

At the beginning of each control cycle (lines 1–3), the UAV swarm initializes the current topology $\mathbf{X}_{\text{current}}$ and loads the candidate set $\mathcal{C}$ with two weighting parameters w_{perf} and w_{life}, which balance throughput and energy. During each iteration (lines 4–10), the UAVs first collect real-time network metrics, including instantaneous $\text{SINR}_{jk}(t)$ for all links and the remaining energy $E_l(t)$ for each node. For every

candidate topology $\mathbf{X}_i \in \mathcal{C}$, a utility score is computed as follow:

$$S_i(t) = w_{perf} \sum_{(j,k)\in\mathbf{X}_i} \log_2(1 + \mathrm{SINR}_{jk}(t)) + w_{life} \min_{l\in\mathcal{V}(\mathbf{X}_i)} \left(\frac{E_l(t)}{E_{\mathrm{init}}} \right) \tag{4.6}$$

where the first term reflects short-term communication performance and the second term represents normalized network endurance. After computing the utility for all candidates, the topology with the highest score is selected:

$$\mathbf{X}_{\mathrm{sel}} = \arg\max_{\mathbf{X}_i\in\mathcal{C}} S_i(t). \tag{4.7}$$

If the selected topology $\mathbf{X}_{\mathrm{sel}}$ differs from the currently deployed one (lines 11–14), a reconfiguration command is broadcast to all UAVs to update their active links accordingly. This evaluation and switching strategy allows the swarm to react quickly to variations in interference, mobility, or energy depletion without solving a new optimization problem in flight. The loop continues until a termination condition is satisfied, such as mission completion or network disconnection.

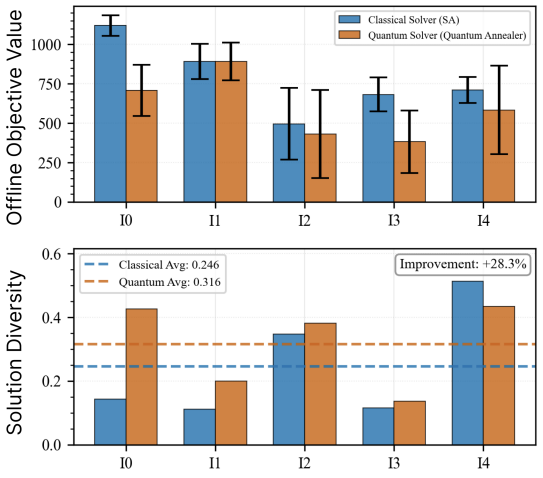
4.3.4 Compatible Deployment with SDN/O-RAN

To ensure deployability, our framework follows the SDN separation of the control plane and user (data) plane. In this model, the ground controller maintains a global view and offloads the computationally intensive QUBO optimization to a quantum annealer. Meanwhile, the UAVs perform only lightweight execution and local measurements. The same workflow maps cleanly to O-RAN. In the *Non-RT RIC* (non-real-time RAN Intelligent Controller, $>$ 1 sec timescale) hosted in the SMO/cloud, rApps periodically generate and refresh a diverse set of candidate topologies and issue policies or intent via the A1 interface. In the *Near-RT RIC* (near-real-time RIC, 0.01-1 sec control loops), edge xApps consume real-time link/SINR feedback over the E2 interface and rapidly select or switch among those candidates. This architecture aligns with the standard O-RAN functional split, which defines the Radio Unit (RU), Distributed Unit (DU), and Centralized Unit (CU). These units are interconnected by the fronthaul, midhaul, and backhaul transport networks. This hierarchical cloud-edge design facilitates the primary goal of our framework: heavy QUBO computation is offloaded to the cloud, while the UAVs perform only low-cost, seconds-scale selection and switching in practice.

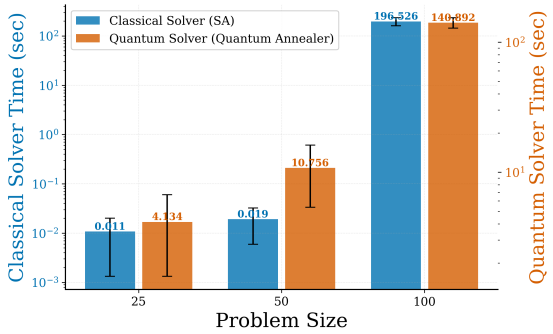
4.4 Experimental Evaluation

The proposed two-stage framework is evaluated under a controlled and fully reproducible setup. Unless otherwise stated, all results are averaged over 20 independent initial deployments and mobility traces per setting. The simulation horizon is 30 seconds with a 1 second time step. Classical optimization is performed using the D-Wave simulated annealing solver on an Apple M4 Pro CPU, and quantum solutions are obtained from a Qboson coherent Ising machine (CPQC-550) configured with a fixed anneal schedule for fair comparison across runs [24]. Throughout this section, Stage 1 refers to offline candidate set generation and Stage 2 refers to online real-time selection. The baseline is a single optimization model that computes one topology at $t = 0$ using the simulated annealing solver, maximizing total throughput without considering network fragility or load imbalance.

4.4.1 Solution Quality and Diversity Comparison



(a) Offline objective value and solution diversity between QA and SA under five UAV application scenarios, I0 through I4.



(b) Runtime scaling of a classical solver and a quantum annealer for $N = 25, 50, 100$ (log–log scale). Error bars show standard deviation across runs.

Figure 4.2: Comparison of QA and SA in Stage 1: (a) solution quality and diversity across five traffic centralization scenarios; (b) runtime scaling with network size.

To evaluate the effectiveness of QA in generating high-quality and diverse topologies under the same time budget, five UAV application scenarios are designed that differ in their levels of traffic centralization. Each scenario represents a UAV topology

network with distinct communication patterns, ranging from evenly distributed flows to highly centralized control, altering the network's betweenness centrality distribution from 0.2 to 1.0 [61]. Each scenario was repeated three times, and in each run, both QA and SA were allowed to generate ten topologies within an equal computation window. The comparison is presented in Figure 4.2a. Two metrics are used for evaluation. The first is the offline objective value, which refers to the QUBO objective value of each obtained topology, defined as a weighted combination of total throughput and network fragility. A lower value indicates a topology with higher aggregate throughput and more balanced traffic distribution, thus representing better communication performance. The second is solution diversity, defined as the average pairwise Hamming distance among all generated topologies, where a higher value reflects richer structural diversity.

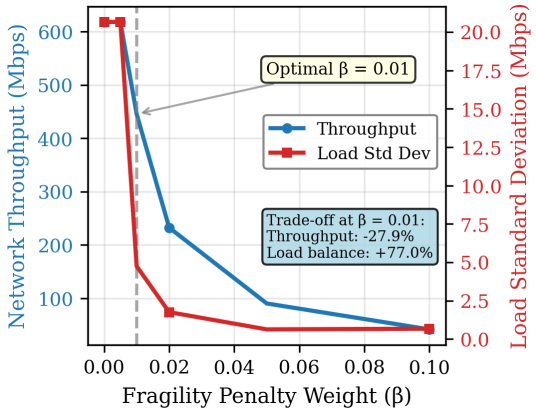
As shown in Figure 4.2a, QA consistently achieves lower offline objective values and higher diversity than SA. Quantitatively, QA improves the average solution diversity by 28.3% and reduces the objective value by 5.15% compared to SA under identical runtime conditions. As discussed in Section 2.4, the sampling behavior of QA differs structurally from that of classical stochastic solvers, which tend to converge repeatedly to similar locally optimal regions. This difference in exploration mechanism enables QA to produce a larger number of high-quality yet structurally distinct candidate topologies within the same computation window. The resulting diversity advantage yields a richer offline candidate pool for the subsequent online selection stage, enhancing adaptability and overall network resilience in dynamic UAV communication environments.

4.4.2 Quantum versus Classical Solver Performance

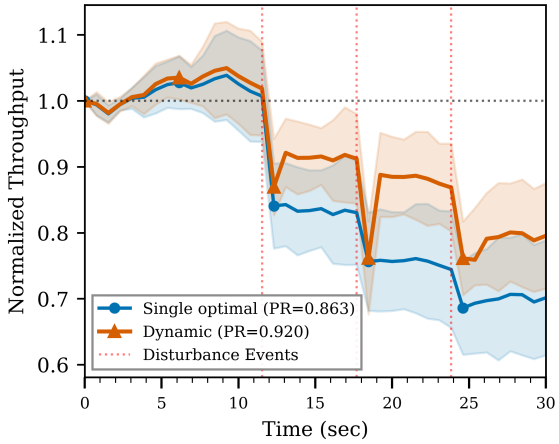
To examine the computational scalability of the QUBO formulation in Stage 1, the SA solver is compared with a quantum annealer across $N \in \{25, 50, 100\}$ (Figure 4.2b). For $N = 25$, the classical SA solver is faster (median 0.011 seconds), which is expected for small instances. However, its runtime increases sharply and reaches 196.5 seconds at $N = 100$. The quantum annealer shows a more gradual increase, from 4.34 seconds at $N = 25$ to 140.9 seconds at $N = 100$, with tight error bars indicating consistent wall-clock behavior across instances. This consistency is important for planning large offline batches, as it enables predictable completion of a structurally diverse candidate set. Although classical solvers are faster for small UAV swarms, the quantum annealer

exhibits more consistent runtime behavior and produces more diverse solutions as the network size increases.

4.4.3 Performance-Fragility Trade-off Analysis



(a) Trade-off between network throughput and load balance as a function of the fragility weight β. Throughput (blue, left axis) decreases as β increases, while load balance (red, right axis), measured by the standard deviation of nodal load, improves accordingly.



(b) Normalized throughput over 30 seconds under mobility with mission disturbances. The dashed line at $y = 1$ marks the normalization reference, and shaded bands show 95% confidence intervals over 20 runs.

Figure 4.3: Offline parameter selection and dynamic performance: (a) throughput versus load balance trade-off as β varies; (b) performance retention comparison between the single optimization model and the dynamic two-stage framework.

Figure 4.3a quantifies how β balances performance and robustness. As β increases, the hub penalty strengthens and both throughput and load imbalance decrease. A clear drop appears near $\beta = 0.01$. At this point, the standard deviation of the load improves by 77.0%, while the initial throughput experiences a reduction of 27.9%. Beyond $\beta = 0.01$, additional robustness gains are small, but the throughput drops more steeply. Therefore, $\beta = 0.01$ is selected for the generation of Stage 1 candidates, as it provides an efficient compromise between structural integrity and initial performance.

4.4.4 Dynamic Performance Evaluation

The single optimization model is compared with the proposed dynamic two-stage framework over a 30 second mobility window with a 1 second step. A standard air-to-air channel with standard shadowing is assumed, and link capacities are mapped to network throughput under time-division multiple access scheduling subject to external environmental interference. To reflect realistic UAV operations, three disturbance events are injected at approximately 12, 19, and 25 seconds, modeled as brief link outages caused by maneuvering or blockage. The single optimization model computes one fixed topology at $t = 0$ by maximizing total throughput. The dynamic two-stage framework prepares a portfolio of 10 offline candidate topologies and performs lightweight periodic selection with hysteresis and a minimum dwell time, while modeling a short outage for topology switching.

The primary metric is performance retention (PR), defined as $\mathrm{PR} = \frac{1}{T}\sum_{t=1}^{T}\frac{\mathrm{thr}(t)}{\mathrm{thr}(0)}$, the time average of throughput normalized to each method's own value at $t = 0$. As shown in Figure 4.3b, the dynamic two-stage framework attains $\mathrm{PR} = 0.920$, outperforming the single optimization model at $\mathrm{PR} = 0.863$, which corresponds to a relative improvement of approximately 6.6%. The framework also recovers faster after mission disturbances and exhibits lower variability, confirming the robustness gained from offline structural diversity combined with online selection.

4.5 Summary

This chapter presented a two-stage framework to enhance the temporal robustness of communication topologies in dynamic UAV networks. By introducing a fragility penalty into a quantum-compliant QUBO optimization model, we generate inherently resilient decentralized network structures. The offline generation of a diverse topology set, combined with a lightweight online selection mechanism, provides a practical and effective way to achieve sustained network performance without costly in-flight re-optimization. This hybrid approach effectively balances strategic planning with tactical agility, offering a promising solution for robust control in next-generation UAV swarm systems. In future work, we will address scalability and latency via a hierarchical hybrid scheme where classical graph partitioning decomposes large networks exceeding quantum hardware limits, allowing our QUBO-based model to optimize intra-swarm topology while classical methods refine inter-swarm coordination.

Chapter 5

Adaptive Microservice Chain Scheduling in the Cloud-Edge Continuum

5.1 Introduction

This chapter investigates the microservice chain scheduling problem in the cloud-edge continuum. It corresponds to Stage 3 in Table 1.1, namely discrete-continuous co-design. Chapter 4 overcomes the limitation of fixed single-instance workflows in time-varying environments by separating offline candidate generation from online lightweight selection. However, the decision output of that chapter remains entirely discrete and does not involve continuous resource allocation. In practical microservice systems, a scheduler must determine not only the execution priority of tasks but also the continuous execution rate allocated to each task under multidimensional resource capacity constraints. QUBO natively supports only binary variables and cannot directly express such mixed decision requirements. Building on the preceding work, this chapter constructs a mixed decision architecture that couples discrete ranking with continuous rate allocation within a single control loop, thereby addressing Research Gap G3, namely the limited decision expressiveness of existing QUBO workflows. The content of this chapter is based on [23] and has been extended in experimental analysis and methodological discussion.

As the computation capability of personal and edge devices grows, latency-sensitive microservices can be deployed on edge nodes close to the user to improve the qual-

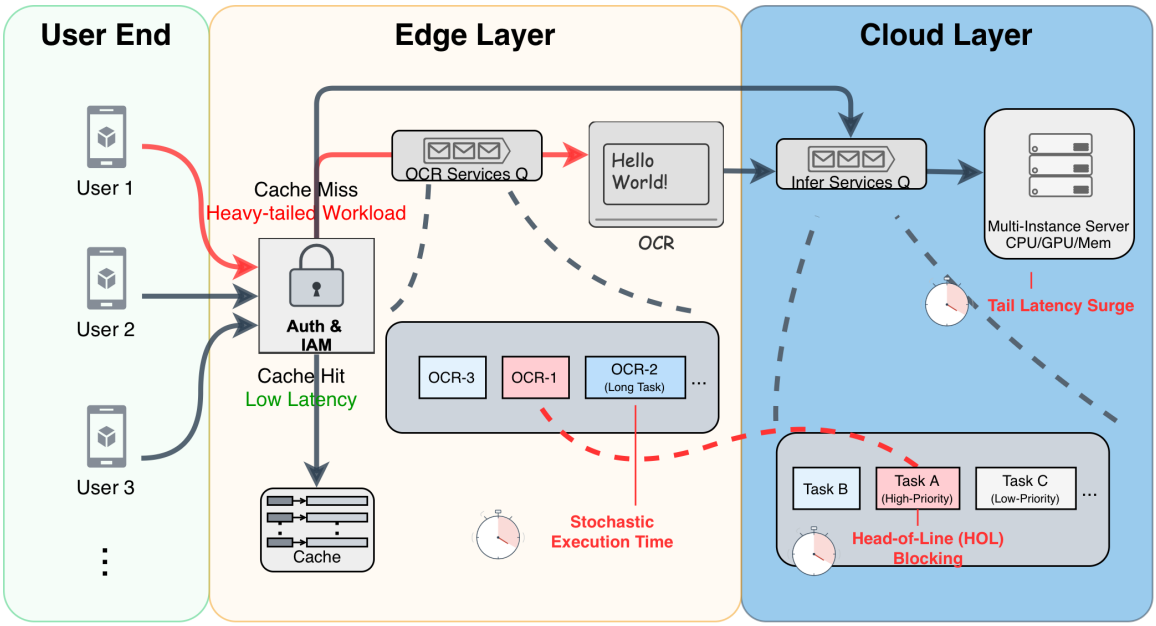


Figure 5.1: End-to-end request paths and node-local scheduling bottlenecks in a microservice-based user-facing application.

ity of service [6]. In interactive online services, microservices are typically composed as directed acyclic graphs that process end-to-end requests [67]. Under this architecture, performance bottlenecks often concentrate at node-local runtime scheduling [68]. Concurrent microservice instances from multiple workflows compete for multidimensional computing resources on the same physical node [69]. This competition is a key driver of end-to-end tail latency and service-level objective violations. In the cloud-edge continuum, microservice execution times are highly stochastic. Input fluctuations and multi-tenant interference can produce heavy-tailed latency distributions and trigger head-of-line blocking. Meanwhile, service meshes provide traffic management and observability, but their data-plane proxies may also introduce additional overhead and amplify latency variability [70]. Consequently, a scheduler must handle both discrete priority decisions and continuous multi-resource allocation constraints within millisecond-scale budgets, while remaining robust to prediction errors.

Figure 5.1 depicts a microservice-based application in the cloud-edge continuum and its end-to-end request paths. Requests arrive at edge ingress, where authentication is performed followed by a cache lookup. On a cache hit, the response is returned directly from the edge cache. On a cache miss, the request enters the edge OCR microservice for text extraction and is then forwarded to the cloud-side inference pipeline. These processing stages create two contention points: the edge OCR

queue and the cloud inference queue. Online scheduling decisions at these two nodes directly shape end-to-end tail latency. Heavy-tailed service times and arrival rate fluctuations caused by cache hit-miss traffic splitting can amplify head-of-line blocking effects. A few long-tail tasks can therefore inflate queueing delays even when aggregate resources are sufficient, leading to service quality violations under bursts.

To tractably solve this mixed-integer optimization problem, this chapter proposes a quantum-inspired adaptive robust scheduling framework called Q-GARS. The key idea is to decompose the node-level scheduling decision into two phases. In the first phase, the priority ranking of ready microservice instances is encoded as a QUBO model. An annealing-type metaheuristic algorithm generates a high-quality discrete ranking prior under strict time budgets. In the second phase, conditioned on the ranking, the system computes continuous execution rates through network utility maximization [27]. This yields a resource slicing strategy that satisfies multidimensional capacity constraints. Because the ranking prior relies on runtime estimates, predictions may fail under severe volatility. The chapter adopts an adaptive robust execution mechanism that treats the prior policy as an untrustworthy expert [71, 72] and competes it in parallel with a stable robust baseline policy. The system adjusts the trust in the prior online via an exponential weight update and ultimately executes an adaptive convex combination of the two. This design preserves empirical performance gains when predictions are accurate and provides a provable performance guarantee relative to the baseline when predictions degrade.

The specific contributions of this chapter are as follows. First, it designs a decoupled node-level scheduling architecture that maps QUBO-based discrete rankings to continuous execution rates satisfying resource constraints. The ranking phase controls the problem size to $O(K^2)$ binary variables through permutation encoding and dynamic penalty coefficient calculation. The rate allocation phase solves continuous resource slicing under the given ranking via network utility maximization. This two-phase architecture directly realizes the coordination between QUBO as an intermediate abstraction layer and classical continuous control discussed in Chapter 2. Second, it proposes a robust adaptive execution mechanism. By mixing the prior policy and a robust baseline online via exponential weights, the mechanism provides a sublinear regret guarantee relative to the robust baseline. Under non-stationary shocks, this mechanism limits the 95th percentile queue backlog peak to 20% to 30% of the baseline. Third, it conducts a systematic evaluation through large-scale discrete-event simulations with 4096 parallel runs. The framework reduces the average weighted

completion time by 2.1%, with improvements reaching 16.8% in complex topologies. In high-volatility scenarios, the system effectively suppresses tail latency and ensures rapid recovery.

The remainder of this chapter is organized as follows. Section 5.2 reviews related work. Section 5.3 describes the system model and the global optimization objective. Section 5.4 presents the node-level QUBO formulation, the continuous resource allocation mechanism, and the adaptive execution design. Section 5.5 conducts the experimental evaluation and analyzes the results. Section 5.6 summarizes the chapter.

5.2 Related Work

In the traditional operations research domain, microservice scheduling and service chaining problems are often formulated as ILP or MILP models. The general methodological foundations of such formulations are discussed in Section 2.2. Harutyunyan et al. formulate a joint user association, service function chain placement, and resource allocation problem as an ILP, followed by a heuristic to address scalability [73]. Kiji et al. formulate multicast service chaining as an ILP that jointly decides VNF placement and routing to minimize deployment and link-usage costs [74]. Beyond service chaining, ILP is also a standard tool for delay-sensitive distributed server allocation and fault-tolerant provisioning. Exact formulations are often accompanied by approximation or heuristic methods to meet practical requirements [75, 76]. These methods provide a principled way to encode global coupling constraints and yield optimal solutions when solved to optimality, but they often require heuristics, relaxations, or approximation algorithms to satisfy practical scalability and online time budgets.

To overcome the computational bottleneck of exact ILP/MILP solvers, researchers have begun adopting emerging computing paradigms such as quantum annealing (QA) and simulated quantum annealing (SQA) for combinatorial subproblems in scheduling. These paradigms require the problem to be transformed into a QUBO representation, whose general conversion procedure is reviewed in Section 2.3. Zhang et al. study the job-shop scheduling problem (JSP) and propose a rank-guided large neighborhood search method that hybridizes a QUBO model with constraint programming, achieving results competitive with state-of-the-art CP approaches on medium-scale JSP instances [20]. The rank variable encoding used in that work provides a reference for subsequent permutation-based QUBO formulations. Pérez Armas et al.

study the resource-constrained project scheduling problem (RCPSP), systematically compare 12 MILP formulations, convert the most qubit-efficient one into a QUBO with explicit penalty terms for hard constraints, and evaluate it on the D-Wave Advantage 6.3 quantum annealer [77]. Although these studies are conducted on canonical operations research benchmarks, the core structures captured by JSP and RCPSP, namely precedence constraints and shared multi-resource capacity constraints, also arise in microservice workflow scheduling. This supports using QUBO/QA as a time-budgeted combinatorial search primitive for resource-constrained scheduling.

From a cloud systems perspective, recent microservice scheduling frameworks emphasize end-to-end QoS under communication overhead and multi-tenant resource contention. Fu et al. propose Nautilus, which combines a communication-aware microservice mapper with contention-aware per-node resource management and runtime migration mechanisms to improve resource efficiency while protecting tail-latency objectives under interference [6]. These systems demonstrate the necessity of fast, interference-aware control loops inside the cloud-edge continuum. However, their node-level controllers typically rely on learned policies or heuristic rules rather than an explicit, time-budgeted combinatorial search primitive that can rapidly explore priority permutations under strict millisecond deadlines. This gap constitutes the direct motivation for the decoupled design in this chapter, which uses QUBO/SQA to generate a lightweight structural prior for ordering decisions while retaining continuous multi-resource allocation and robust safeguarding to handle stochastic latencies. The methodological positioning of such discrete-continuous co-design is discussed in Section 2.5.

5.3 System Model

This section formalizes the microservice scheduling problem within the cloud-edge architecture as a discrete-time control model. The system is managed through a two-timescale architecture. The global orchestrator handles service placement logic over coarse-grained periods. The node-level scheduler executes local control at millisecond-scale decision epochs. At each discrete decision epoch, the system observes and extracts the set of ready tasks that have satisfied their upstream directed acyclic graph dependencies as the current state. The control action of the system is to compute the instantaneous resource allocation rates subject to the multidimensional capacity limits of the heterogeneous nodes. The global optimization objective of the system is

to minimize the total weighted completion time of all microservice workflows. Given the large scale of active workflows and microservices, the global objective function is computationally intractable to solve directly within a single decision epoch. The node-level controller utilizes local queueing delay and resource contention penalties as a tractable surrogate for this global objective to make real-time decisions. Table 5.1 summarizes the core mathematical notations used throughout this chapter.

Table 5.1: Summary of Core Sets, States, and Decision Variables

Sets and System Parameters	
Notation	Description
$\mathcal{J}$	Set of user request workflows, indexed by j
$\mathcal{M}$	Set of heterogeneous compute nodes, indexed by m
C_m	Multidimensional computing resource capacity vector $C_m \in \mathbb{R}_+^d$
ω_j	Service level agreement weight of workflow j
State and Control Variables	
Notation	Description
t	Discrete decision epoch
$S_m(t)$	Node-local active ready microservice set at decision epoch t
$\mathbf{r}_v(t)$	Multidimensional execution rate vector allocated to microservice v
$\mathbf{r}^*(t)$	Final deterministic mixed execution rate vector
$\lambda(t)$	Adaptive trust parameter for the online learning framework
Intermediate Algorithmic Variables (Section 5.4)	
Notation	Description
r	Discrete execution rank index for microservices
$c_{v,r}$	Base linear proxy cost of assigning microservice v to rank r
$\theta_v(t)$	Continuous computing resource allocation share for microservice v

5.3.1 Infrastructure and System State

The physical cloud-edge infrastructure is modeled as a set of heterogeneous compute nodes $\mathcal{M}$. Each node $m \in \mathcal{M}$ possesses a multidimensional resource capacity vector $C_m \in \mathbb{R}_+^d$, covering hardware limits such as CPU, memory, and network bandwidth. These limits mathematically define a multidimensional packing polytope, which establishes the physical boundary for local node scheduling. User service requests are

represented as a set of workflows $\mathcal{J}$. Each workflow $j \in \mathcal{J}$ is mapped to a directed acyclic graph $G_j = (V_j, E_j)$. The vertices $v \in V_j$ represent specific microservice tasks. The directed edges $(u, v) \in E_j$ represent strict upstream data dependencies between tasks. The execution node assigned to task v by the global orchestrator is denoted $\mu(j, v) \in \mathcal{M}$. Because the actual execution latencies of microservices are non-clairvoyant, the node-level scheduler acts solely based on the observable state at discrete decision epochs t. A key component of the observable local state is the node-local ready set $S_m(t)$. The system telemetry state simultaneously includes the queue backlog $q_v(t)$ and the observed enqueue rate $a_v(t)$ for task v. Task v is included in $S_m(t)$ and becomes eligible to compete for available intra-node resources only when all its upstream predecessor tasks u have completed and $\mu(j, v) = m$.

5.3.2 Control Action and Surrogate Objective

After the system state is defined, the control action and the optimization objective are further specified. At each decision epoch t, the scheduler on node m allocates an instantaneous multidimensional execution rate vector $\mathbf{r}_v(t) \in \mathbb{R}_+^d$ to each task v in the local active ready set $S_m(t)$. The aggregate resource allocation for all concurrent tasks must satisfy the multidimensional packing polytope constraint. This polytope is mathematically defined as a time-dependent set of decision vectors: $\mathcal{P}_m(t) = \{\{\mathbf{r}_v(t)\}_{v \in S_m(t)} : \mathbf{r}_v(t) \geq 0, \sum_{v \in S_m(t)} \mathbf{r}_v(t) \leq C_m\}$. The global optimization objective of the system is to minimize the total weighted completion time of all workflows, formulated as $\min \sum_{j \in \mathcal{J}} \omega_j C_j$, where C_j represents the end-to-end completion time of the final microservice task in workflow j. Because of the non-clairvoyant nature of the real physical environment, the actual execution latency p_v of a microservice is stochastic and unknown. The system can only observe this latency upon task completion. This makes the global objective function intractable to solve directly within a single decision epoch. To obtain an implementable online scheduler, the chapter approximates the global weighted-completion-time objective with a node-local surrogate that can be evaluated at each millisecond-scale decision epoch.

Intuitively, the scheduler should (i) prioritize tasks with large backlogs to reduce queueing delay and tail latency, and (ii) avoid co-running strongly interfering tasks to mitigate resource contention. Accordingly, the surrogate objective takes the form

$$\min \sum_{v \in S_m(t)} \phi(q_v(t), \theta_v(t)) \; + \sum_{v<u} \psi(\mathbf{r}_v(t), \mathbf{r}_u(t)) \,, \tag{5.1}$$

where ϕ quantifies per-task delay pressure. Under the scalar resource share θ_v introduced in Section 5.4.3, ϕ is instantiated as

$$\phi(q_v, \theta_v) \;=\; \frac{q_v}{\theta_v}, \tag{5.2}$$

which is strictly decreasing and strictly convex in θ_v for $\theta_v > 0$. This convexity property is used in the theoretical guarantee of Section 5.4.4. The term ψ quantifies pairwise contention for shared resources. These two components serve different architectural phases of the framework. In Section 5.4.2, ϕ and ψ are jointly encoded into a permutation-based QUBO model: ϕ induces the linear ranking cost $c_{v,r}$ that captures the urgency of placing v at rank r, and ψ induces the pairwise interference coefficient $Q_{v,u}$ modulated by a distance-decaying kernel $g(\cdot)$. In the adaptive execution phase of Section 5.4.4, the permutation has already been fixed and the contribution of ψ has been absorbed into the discrete ranking result. The real-time shadow loss is therefore computed from the aggregate ϕ values only, which preserves the convexity of the per-epoch cost with respect to the rate vector.

5.4 Quantum-Inspired and Adaptive Robust Scheduling Framework

This section presents an online scheduling framework to approximately solve the local surrogate objective introduced in Section 5.3.2 under millisecond-level decision budgets. Figure 5.2 illustrates the overall architecture of this framework. The system uses a local quantum annealing simulator driven by SQA algorithms. Because the framework models the scheduling problem as a standardized QUBO via the simulator, it is entirely decoupled from the underlying solver. This design is consistent with the positioning of QUBO as an intermediate abstraction layer discussed in Section 2.3, allowing the system to seamlessly switch to real physical quantum annealing devices once edge-quantum hardware matures or communication latency bottlenecks are resolved. To achieve this goal, the framework adopts a four-phase design: local state extraction and dimensionality reduction, quantum-inspired structural prior generation, continuous rate allocation, and adaptive robust execution.

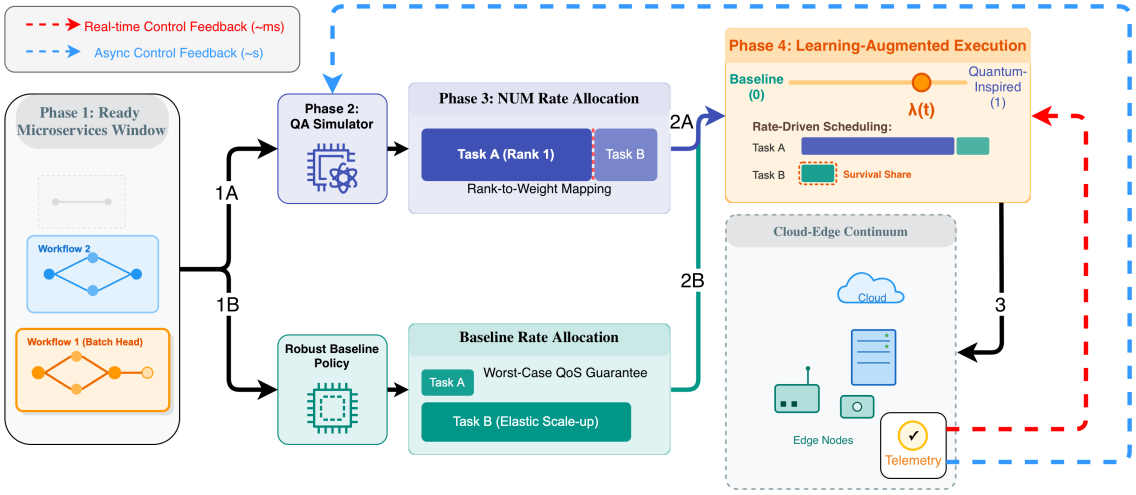


Figure 5.2: Overview of the closed-loop adaptive scheduling framework integrating SQA and real-time feedback control.

5.4.1 State Extraction and Dimensionality Reduction

In the first phase, at each decision epoch t, the node-level controller extracts the local active ready set $S_m(t)$ from the workflow directed acyclic graphs, as illustrated by the leftmost module in Figure 5.2. Since microservices in $S_m(t)$ have satisfied all hard upstream dependencies, they are eligible to compete for intra-node computing resources. Directly optimizing the continuous multidimensional execution rate vector $\mathbf{r}_v(t)$ for all ready microservices is prohibitive due to the combinatorial structure induced by contention and priority interactions. To limit the combinatorial search space, the system retains only the top K candidate microservices according to a lightweight priority score, using the arrival order as a tie-breaker. This forms a dynamic active window $\tilde{S}_m(t)$ and yields $O(K^2)$ binary variables in the subsequent permutation-indexed QUBO encoding.

5.4.2 QUBO Formulation and Rank Generation

In the second phase, the microservice prioritization is mapped into a QUBO model. Binary decision variables $x_{v,r} \in \{0, 1\}$ are defined for the active window $\tilde{S}_m(t)$. This variable equals 1 if microservice $v \in \tilde{S}_m(t)$ is assigned to the r-th execution rank and equals 0 otherwise, where $1 \leq r \leq K$. The complete objective function consists of a cost part and a penalty part: $H(\mathbf{x}) = H_{\text{cost}}(\mathbf{x}) + H_{\text{pen}}(\mathbf{x})$.

The cost function captures the linear delay penalties and soft non-linear computing resource contention among microservices:

$$H_{\text{cost}}(\mathbf{x}) = \sum_{v \in \tilde{S}_m(t)} \sum_{r=1}^{K} c_{v,r} x_{v,r} + \sum_{\substack{v,u \in \tilde{S}_m(t) \\ v \neq u}} \sum_{1 \leq s < r \leq K} g(r-s)\, Q_{v,u}\, x_{v,r}\, x_{u,s} \tag{5.3}$$

The term $c_{v,r}$ represents the base linear proxy cost of assigning microservice v to rank r. It is induced by the delay-pressure function ϕ (Section 5.3.2) under the rank substitution and is defined as $c_{v,r} = q_v(t) \cdot r$, where $q_v(t)$ is the current queue backlog of v. This definition captures a direct scheduling intuition: tasks with larger backlogs incur higher cost when placed at lower priority rank positions. Because explicit concurrent execution overlap is difficult to model using pure ordinal variables, the chapter introduces $Q_{v,u}$ as an order-induced interference proxy that captures the contention intensity between two tasks for shared resources. It is defined as $Q_{v,u} = q_v(t)\, q_u(t) / (\max_{w \in \tilde{S}_m(t)} q_w(t))^2$, which ensures $Q_{v,u} \in [0, 1]$. The distance decay kernel is defined as $g(\Delta) = \exp(-\beta\, \Delta)$, where $\beta > 0$ is a decay rate parameter and $\Delta = r - s \geq 1$. This kernel satisfies $g(\Delta) \in (0, 1)$ and decreases with rank separation. Its physical interpretation is that the externality of resource contention imposed on microservice v at rank r by a competing microservice u at a higher-priority rank s decays as the rank gap between them increases.

To convert the problem into an unconstrained form, the penalty function enforces a strict one-to-one mapping between tasks and ranks via quadratic terms:

$$H_{\text{pen}}(\mathbf{x}) = A \sum_{v \in \tilde{S}_m(t)} \left(\sum_{r=1}^{K} x_{v,r} - 1 \right)^2 + B \sum_{r=1}^{K} \left(\sum_{v \in \tilde{S}_m(t)} x_{v,r} - 1 \right)^2 \tag{5.4}$$

This quadratic penalty encoding of permutation constraints follows the general QUBO constraint embedding procedure introduced in Section 2.3. A and B are penalty coefficients. Static empirical lower bounds often generate excessively large penalty weights that overwhelm the underlying cost optimization objective and cause the solver to stall in local optima. To resolve this issue within millisecond budgets, the chapter adopts a dynamic maximum marginal cost method to automatically set the penalty coefficients. Given that the distance-decaying kernel satisfies $0 \leq g(r-s) \leq 1$, it can be safely omitted to form a conservative upper bound for quadratic interactions. During the construction of the cost matrix, the system calculates the maximum local objective increment that can occur when any single microservice is assigned. The

penalty coefficients are set as $A = B = \max_{v \in \tilde{S}_m(t)}(\max_r c_{v,r} + \sum_{u \neq v} Q_{v,u})$. This choice serves as a conservative sufficient condition. It ensures that the penalty incurred by any single-variable move violating the constraints cannot be offset by an improvement in the cost Hamiltonian H_{cost}. This dynamic calculation strictly bounds the time complexity to $O(K^2)$. It generates adaptive penalty bounds that ensure feasibility dominance while avoiding the additional latency imposed on the control loop by complex matrix parsing operations.

5.4.3 Continuous Computing Resource Allocation

In the third phase, the system translates the discrete microservice permutation into continuous resource allocation. Because the data dependency of microservices and resource interference exhibit strong non-convex combinatorial optimization properties, directly solving for continuous rates leads to severe local optima. Therefore, the chapter adopts the decoupled scheduling principles common in modern operating systems: the system first determines the execution rank of the microservices and subsequently partitions the continuous computing resources.

The system translates the discrete ordering into a continuous relative weight $\tilde{w}_v(t)$. This soft isolation mechanism eliminates the system overhead incurred by hard preemption such as context switching [78]. To ensure deterministic weight assignment and strictly favor critical microservices, the chapter adopts an exponential decay mapping strategy. For a microservice $v \in \tilde{S}_m(t)$ assigned to the execution rank r ($r \in \{1, 2, \ldots, K\}$), its weight is defined as $\tilde{w}_v(t) = \gamma^{K-r}$, where $\gamma > 1$ is a tunable decay factor. For ready microservices that do not fall into the truncated active window, namely $v \in S_m(t) \setminus \tilde{S}_m(t)$, the system assigns a minimal base weight $0 < \epsilon \ll 1$ to prevent complete starvation without disrupting the active window's dominance.

Finally, to obtain a closed-form solution that satisfies multidimensional resource physical constraints under millisecond budgets, resources are allocated based on the classical network utility maximization theory [79]. The computing resource allocation share of microservice v at time t is defined as a continuous scalar $\theta_v(t) \in (0, 1]$. To avoid complex multi-commodity flow calculations under strict millisecond latency budgets, this scalar is mapped uniformly across all hardware dimensions as a normalized computing resource share. Let $\mathbf{C}_m$ denote the multidimensional capacity vector of machine m; the execution rate vector of microservice v is formalized as $\mathbf{r}_v^q(t) = \theta_v(t)\mathbf{C}_m$. This implies that the multidimensional physical constraint

$\sum_{v \in S_m(t)} \mathbf{r}_v^q(t) \leq \mathbf{C}_m$ is mathematically equivalent to a simplified scalar constraint $\sum_{v \in S_m(t)} \theta_v(t) \leq 1$. To achieve weighted proportional fairness, a logarithmic utility function is introduced as the optimization objective:

$$\begin{aligned} \max_{\boldsymbol{\theta}(t)} \quad & \sum_{v \in S_m(t)} \tilde{w}_v(t) \log(\theta_v(t)) \\ \text{s.t.} \quad & \sum_{v \in S_m(t)} \theta_v(t) \leq 1, \quad \theta_v(t) > 0, \ \forall v \in S_m(t) \end{aligned} \tag{5.5}$$

The use of logarithmic utility to achieve proportional fairness originates from the foundational work of Kelly et al. [27]. Because the objective function is strictly concave and the constraints form a standard probability simplex, strong duality holds for this optimization problem. By applying the Karush-Kuhn-Tucker conditions, $\theta_v(t) \propto \tilde{w}_v(t)$ and the capacity constraint is tightly binding. This directly yields a closed-form continuous rate allocation scheme:

$$\theta_v^*(t) = \frac{\tilde{w}_v(t)}{\sum_{u \in S_m(t)} \tilde{w}_u(t)} \implies \mathbf{r}_v^q(t) = \left(\frac{\tilde{w}_v(t)}{\sum_{u \in S_m(t)} \tilde{w}_u(t)} \right) \mathbf{C}_m \tag{5.6}$$

This mechanism accurately preserves the relative priority structure implied by the discrete permutation via $\theta_v(t) \propto \tilde{w}_v(t)$. The aggregate normalization ensures that the allocation scheme fully utilizes the node capacity without violating the multidimensional strict boundary $\mathcal{P}_m(t)$, achieving safe and instantaneous continuous resource slicing.

5.4.4 Adaptive Robust Execution and Safety Guarantee

The SQA-produced ranking prior can be unreliable under severe runtime volatility. To ensure worst-case robustness, the chapter adopts a learning-augmented execution rule with untrusted predictions [71, 72]. At each decision epoch t, the quantum-guided allocator and a robust baseline allocator are treated as two competing experts, producing feasible rate vectors $\mathbf{r}^q(t)$ and $\mathbf{r}^b(t)$, respectively. A trust weight $\lambda(t) \in [0, 1]$ is maintained for the quantum-guided expert, initialized as $\lambda(1) = 0.5$.

Shadow Loss and Weight Update

To obtain a fast and stable feedback signal within millisecond budgets, a lightweight shadow model computes per-epoch surrogate losses $\hat{L}^q(t)$ and $\hat{L}^b(t)$ for the two ex-

perts. Specifically, $\hat{L}^q(t)$ (respectively $\hat{L}^b(t)$) evaluates only the delay-pressure component $\sum_{v \in S_m(t)} \phi(q_v, \theta_v)$ of the surrogate objective defined in Section 5.3.2 at the rate vector $\mathbf{r}^q(t)$ (respectively $\mathbf{r}^b(t)$). Because the permutation is fixed in Section 5.4.2, the pairwise contention term ψ has been absorbed into the discrete ranking result and does not enter the shadow loss. Each epoch's shadow loss is normalized to $[0, 1]$ by dividing by the maximum single-epoch delay pressure $\max_t \sum_v q_v(t)/\theta_v(t)$.

The trust weight is updated using the standard exponential-weights (Hedge) rule:

$$\lambda(t{+}1) = \frac{\lambda(t) \exp\bigl(-\eta \hat{L}^q(t)\bigr)}{\lambda(t) \exp\bigl(-\eta \hat{L}^q(t)\bigr) + (1{-}\lambda(t)) \exp\bigl(-\eta \hat{L}^b(t)\bigr)}, \tag{5.7}$$

where $\eta > 0$ is a learning rate.

Deterministic Rate Mixing

The node-level scheduler executes a deterministic convex combination of the two feasible allocations:

$$\mathbf{r}^*(t) = \lambda(t)\, \mathbf{r}^q(t) + \bigl(1{-}\lambda(t)\bigr)\, \mathbf{r}^b(t). \tag{5.8}$$

Because both $\mathbf{r}^q(t)$ and $\mathbf{r}^b(t)$ lie in the feasibility polytope $\mathcal{P}_m(t)$ and $\mathcal{P}_m(t)$ is convex, the mixed rate $\mathbf{r}^*(t)$ is also feasible for every $\lambda(t) \in [0, 1]$.

Regret Guarantee

The formal performance guarantee of the adaptive mechanism is now stated.

Assumption 1. *The per-epoch surrogate losses satisfy* $\hat{L}^q(t),\ \hat{L}^b(t) \in [0, 1]$ *for all* t.

Assumption 2. *The per-epoch scheduling cost* $c(\mathbf{r}, t)$ *is convex in the rate vector* $\mathbf{r}$ *for each decision epoch* t.

Because the shadow loss is computed from the delay-pressure component ϕ alone (see Section 5.4.4), Assumption 2 requires only that $\phi(q_v, \theta_v) = q_v/\theta_v$ is convex in θ_v. For $\theta_v > 0$, the second derivative $\mathrm{d}^2\phi/\mathrm{d}\theta_v^2 = 2q_v/\theta_v^3 > 0$, so Assumption 2 holds by construction.

Proposition 1. *Under Assumptions 1 and 2, the cumulative scheduling cost of the mixed policy* $\mathbf{r}^*(t)$ *over a horizon of* T *decision epochs satisfies*

$$\sum_{t=1}^{T} c\bigl(\mathbf{r}^*(t), t\bigr) \;\le\; \min\left\{ \sum_{t=1}^{T} c\bigl(\mathbf{r}^q(t), t\bigr),\ \sum_{t=1}^{T} c\bigl(\mathbf{r}^b(t), t\bigr) \right\} \;+\; \frac{\ln 2}{\eta} + \frac{\eta\, T}{8}\,. \tag{5.9}$$

Proof. Fix an arbitrary epoch t. By Assumption 2 and the definition of convexity,

$$c\big(\mathbf{r}^*(t), t\big) = c\big(\lambda(t)\,\mathbf{r}^q(t) + (1-\lambda(t))\,\mathbf{r}^b(t),\, t\big) \le \lambda(t)\, c\big(\mathbf{r}^q(t), t\big) + (1-\lambda(t))\, c\big(\mathbf{r}^b(t), t\big).$$

The right-hand side is the cost of a randomized choice between two experts with mixture weight $\lambda(t)$, which is exactly the quantity governed by the Hedge algorithm. Summing over $t = 1, \ldots, T$ and applying the two-expert Hedge regret bound [80] yields (5.9). □

The regret term $\frac{\ln 2}{\eta} + \frac{\eta T}{8}$ is minimized at $\eta^* = \sqrt{8 \ln 2 / T}$, yielding $O(\sqrt{T})$ cumulative regret and thus $O(1/\sqrt{T})$ vanishing average regret. In practice, the decision horizon T is not known in advance. The learning rate η is therefore set as a fixed constant calibrated to the expected episode length. Section 5.5 validates empirically that this choice provides effective adaptation across both shock and recovery regimes.

When the quantum-guided prior becomes unreliable, the surrogate loss $\hat{L}^q(t)$ increases relative to $\hat{L}^b(t)$, causing $\lambda(t)$ to decrease via (5.7). The scheduler thereby shifts weight toward the robust baseline without operator intervention. Proposition 1 guarantees that the cost of this mixed policy never exceeds the cost of the better expert by more than $O(\sqrt{T})$ over any horizon, providing a worst-case safeguard against non-stationary prediction failures.

5.5 Experimental Evaluation

To validate the effectiveness of Q-GARS, a discrete-event simulation platform was constructed. To satisfy the millisecond decision budgets established in Section 5.3.2, the evaluation environment was deployed on a high-performance computing cluster equipped with a local NVIDIA A40 GPU. This local coprocessor architecture eliminates cloud communication latency. For the active window size $K \le 100$ defined in Section 5.4.1, the resulting QUBO models contain up to 10,000 binary variables with dense interaction structures from the permutation encoding. The largest current quantum annealing processor, the D-Wave Advantage, has approximately 5000 physical qubits with a Pegasus topology providing 15 connections per qubit [54]. For densely interacting QUBO instances, the number of physical qubits required by minor embedding far exceeds the number of logical variables, making a 10,000 variable instance infeasible for native embedding on current QPU hardware. The experiments therefore employ a local SQA solver, which resolves these models in sub-millisecond

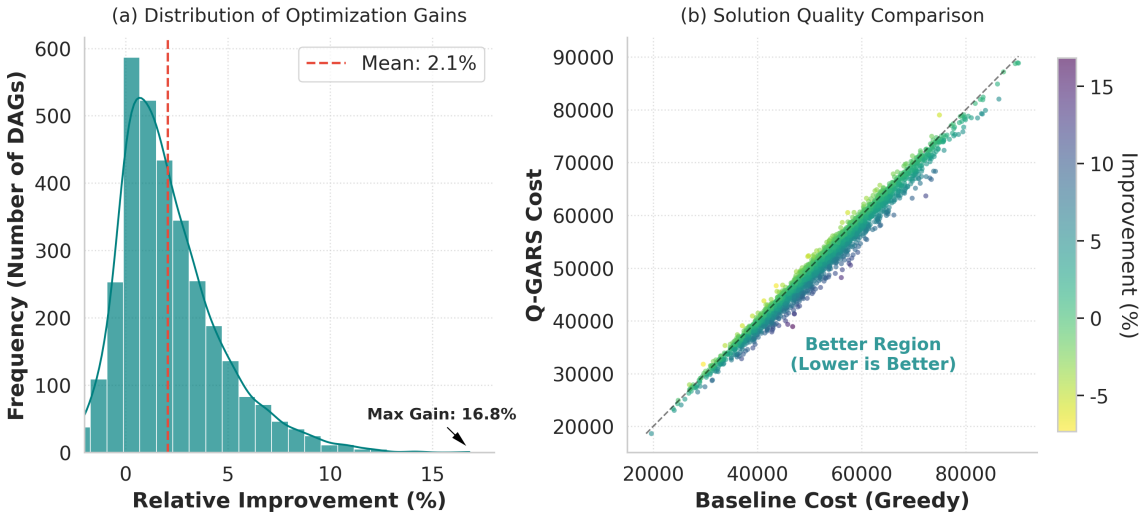


Figure 5.3: (a) Histogram showing the distribution of relative performance improvements over the greedy baseline. (b) Scatter plot comparing solution quality, indicating consistent improvements in high-cost regions.

time. As discussed in Section 5.4, because the framework is decoupled from the underlying solver through a standardized QUBO interface, the system can seamlessly switch to physical quantum annealing devices once future quantum hardware surpasses current qubit count and connectivity limitations, without modifying the framework architecture. The experiments simulated tens of thousands of concurrent workflows using Monte Carlo methods to ensure statistical significance. The evaluation consists of three progressive phases: structural prior gain, worst-case robustness, and system dynamics at scale.

5.5.1 Performance Gain from Structural Prior

The performance improvement brought by the quantum-inspired structural prior is first quantified. A dataset of directed acyclic graphs was generated. The number of nodes per graph follows a uniform distribution between 10 and 50, with a maximum concurrency width ranging from 2 to 8. The sequence generated by Q-GARS was compared against a greedy baseline policy based on the shortest remaining processing time. To ensure a rigorous comparison, both discrete ordering results are fed into the NUM module described in Section 5.4.3 for continuous rate slicing.

Figure 5.3(a) shows the distribution of optimization gains among $N = 10{,}000$ independent samples. The structural prior significantly reduces the global weighted completion time, achieving a 2.1% mean improvement, with the maximum improve-

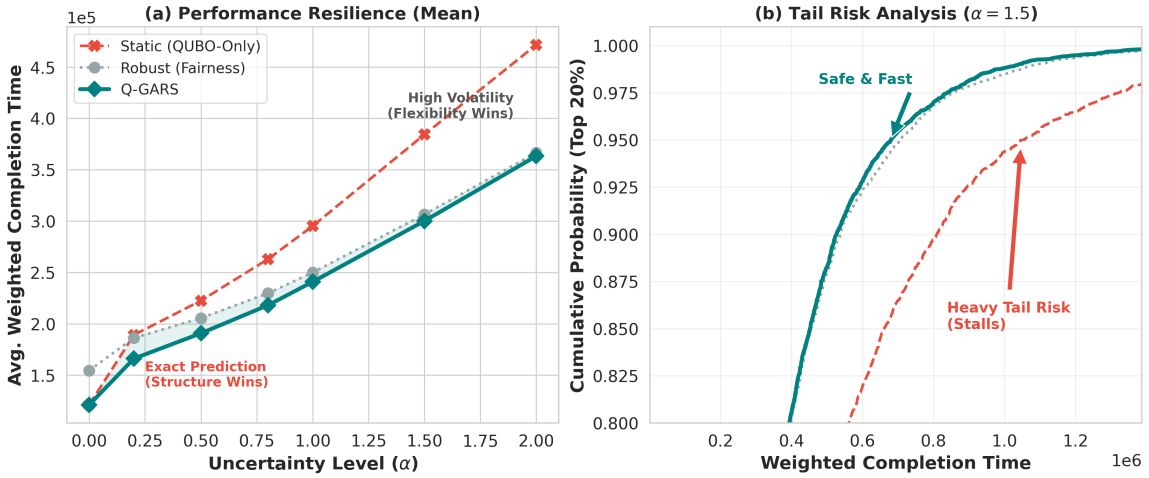


Figure 5.4: (a) Impact of increasing uncertainty level α on average weighted completion time. (b) Cumulative Distribution Function (CDF) of completion times under high volatility ($\alpha = 1.5$), highlighting the mitigation of heavy tail risks.

ment reaching 16.8%. The scatter plot in Figure 5.3(b) further compares the solution quality. As the baseline objective value increases, the solution of Q-GARS consistently remains in the "Better Region" below the diagonal. This demonstrates the solving consistency of the algorithm within complex state spaces.

5.5.2 Resilience Under Stochastic Volatility

In real world, cloud-edge environments exhibit stochastic execution times [69]. In the second phase, the stochastic latency model is parameterized using α to control the magnitude of fluctuations and inject tail latency. Figure 5.4 demonstrates the performance of different scheduling mechanisms regarding the weighted completion time under uncertainty. As shown in Figure 5.4(a), Q-GARS consistently maintains the lowest average weighted completion time at all uncertainty levels. In the near-deterministic regime ($\alpha \approx 0$), the static prior policy performs comparably to Q-GARS. As volatility increases (e.g., $\alpha \geq 0.25$), Q-GARS rapidly establishes a gap and consistently outperforms both the static prior and the robust baseline policy. Even in the high-noise regime, the performance of the robust baseline merely approaches Q-GARS but does not surpass it in the experiments. In a highly volatile setting ($\alpha = 1.5$), the upper-tail cumulative probability distribution in Figure 5.4(b) shows that Q-GARS and the robust baseline significantly shift the worst completion times to the left compared to the static prior. This indicates that the adaptive trust parameter

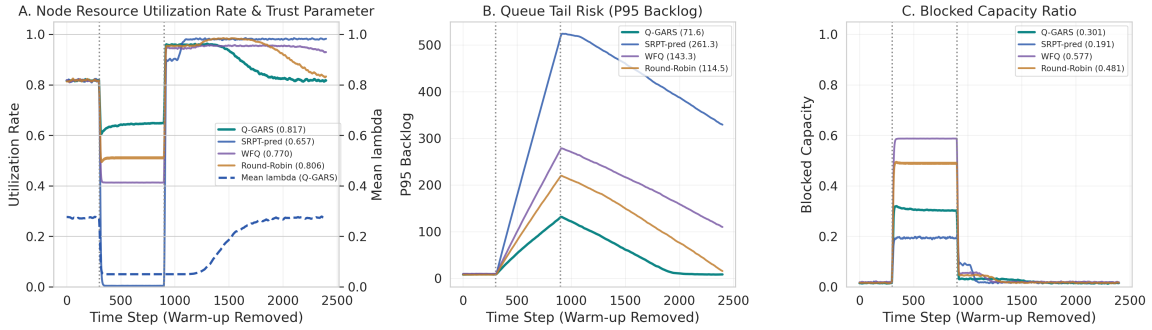


Figure 5.5: System dynamics under a simulated shock interval from time step 300 to 900. (a) The cross-system mean trust parameter $\bar{\lambda}(t)$ drops to trigger the safeguard mode. (b) Q-GARS effectively suppresses the peak surge of the 95th percentile queue backlog. (c) Blocked capacity ratio is minimized and recovers to near-zero levels fastest.

effectively mitigates extreme latency events. Overall, these observations are consistent with the guarantee of Proposition 1. In the low-uncertainty regime, the ranking prior is accurate and Q-GARS tracks the static prior. In the high-uncertainty regime, the prior degrades, $\lambda(t)$ decreases via the Hedge update, and Q-GARS automatically falls back toward the robust baseline. In the intermediate regime, the mixed policy tracks whichever expert performs better. Proposition 1 guarantees that the cumulative cost of the mixed policy exceeds that of the better expert by at most $O(\sqrt{T})$; the crossover behavior of the three curves in Figure 5.4(a) is an empirical manifestation of this theoretical property.

5.5.3 System Dynamics and Shock Recovery

Finally, large-scale system resilience was evaluated through long-horizon simulations of 4096 independent parallel systems. A shock interval from time step 300 to 900 is marked with vertical dashed lines in Figure 5.5. During this period, node failure rates and prediction errors increase sharply.

As shown in Figure 5.5(a), the cross-system mean trust parameter $\bar{\lambda}(t)$ drops rapidly during the shock. This indicates that the shadow loss signal detects prediction degradation. The system moves toward safeguard mode via the exponential weight update rule detailed in Section 5.4.4. After the shock subsides, the trust parameter gradually recovers and the system re-enables the benefits of predictive priors. Figure 5.5(b) reports the 95th percentile queue backlog. Compared with non-adaptive baselines, Q-GARS significantly suppresses the peak backlog surge. Specifically, its

peak is approximately 20% to 30% of the SRPT-pred baseline. Figure 5.5(c) shows the blocked capacity ratio. Q-GARS maintains a lower capacity loss during the shock interval and returns to near-zero levels fastest after recovery. Together, these results empirically demonstrate the graceful degradation and fast recovery capabilities of the framework in non-stationary environments.

5.6 Summary

This chapter investigates the microservice chain scheduling problem in the cloud-edge continuum and proposes the Q-GARS framework, which effectively mitigates head-of-line blocking in heterogeneous computing environments by integrating the combinatorial optimization capabilities of QUBO with the robustness of proportional fairness. The core design is a decoupled two-phase architecture: the first phase encodes node-level priority ranking as a QUBO and generates a discrete ranking prior through an annealing-type solver; the second phase translates the discrete ranking into continuous resource allocation satisfying multidimensional capacity constraints via network utility maximization. The adaptive robust execution mechanism preserves performance gains when the prior is accurate and provides an $O(\sqrt{T})$ regret guarantee relative to the robust baseline when the prior degrades. At the methodological level, this chapter directly addresses Research Gap G3. In Chapters 3 and 4, the decision outputs are entirely discrete. This chapter demonstrates that QUBO can serve as a component within a larger mixed decision loop, where discrete ranking and continuous rate allocation are coordinated within a single control loop. This validates the positioning of QUBO as an intermediate abstraction layer in Section 2.3: QUBO is not an endpoint solution method but an interface connecting combinatorial search with classical continuous control.

In the current experiments, the QUBO scale exceeds the native embedding capacity of existing quantum annealing hardware, and therefore an SQA solver is employed. However, the solver-agnostic design of the framework ensures that when quantum hardware achieves sufficient qubit count and connectivity to satisfy embedding requirements, the system can switch directly to physical quantum annealing devices. Furthermore, the current permutation encoding limits the active window size, and hierarchical decomposition methods that scale to larger workflows represent a direction for future work. The next chapter elevates the problem from domain-specific modeling to the level of automated QUBO formulation and hybrid orchestration.

Chapter 6

Automated QUBO Formulation and Hybrid Orchestration for Combinatorial Optimization

This chapter investigates automated QUBO formulation and scalable hybrid solving for general combinatorial optimization problems. It corresponds to Stage 4 in Table 1.1, namely LLM-driven automated formulation and hybrid orchestration. Chapters 3 – 5 validate the operability, dynamic adaptability, and discrete-continuous coordination capability of QUBO-based workflows in domain-specific settings, but the QUBO construction process in each case still relies on manual modeling. Meanwhile, even when a correct QUBO is obtained, the physical scale of a single quantum device limits direct solving for large problem instances. This chapter addresses Research Gaps G1 and G4 simultaneously. It automates the conversion from a structured MILP to a hardware-aware QUBO through an LLM-driven transformation engine, and it separates the binary master problem from the continuous subproblem through Benders decomposition so that large-scale instances exceeding the capacity of a single quantum processor can be solved within a hybrid framework. The content of this chapter is based on [2] and has been extended in methodological discussion and experimental analysis.

6.1 Introduction

Section 2.3 discusses in detail the conversion pipeline from constrained optimization models to QUBO, including four core steps: binarization, penalty reformulation, slack variable introduction, and variable encoding. Chapters 3 – 5 each execute this pipeline within their respective application settings. The practical experience from these chapters confirms that QUBO modeling is well understood in principle, yet each new problem still requires the analyst to carry out the entire conversion process manually. Specifically, the modeler must identify constraint types and select the corresponding penalty structures, determine the bit-width of slack variables to control the dimensional growth of the QUBO matrix, and calibrate penalty coefficients so that they are large enough to enforce constraint satisfaction without distorting the energy landscape to the point of degrading solver performance [12, 21]. Existing tools such as PyQUBO provide syntax-level support but do not automate constraint analysis or penalty calibration [56]. This situation has a direct analogy in classical computing: it is equivalent to a state in which no compiler exists and users must manually translate high-level logic into low-level machine instructions. This chapter refers to this obstacle as the formulation bottleneck and proposes to eliminate it through LLM-driven automation.

Recent research has made notable progress in using large language models to convert natural language problem descriptions into structured mathematical models automatically. The NL4Opt competition established the entity recognition based autoformulation task [81]. Subsequent systems such as OptiMUS built end-to-end LLM agent pipelines that generate solvable MILP code directly from natural language descriptions [82]. LLMOPT and ORLM further introduced fine-tuning methods on open-source models to improve modeling accuracy [28, 83]. However, these efforts focus on the conversion stage from natural language to MILP. Their output is an MILP model that can be processed directly by classical solvers such as Gurobi. The subsequent conversion from MILP to a hardware-compatible QUBO still requires manual effort. At the QUBO generation level, PyQUBO provides a syntactic interface that compiles Python-defined constraints into QUBO matrices [56]. AutoQUBO employs a data-driven interpolation method to extract QUBO coefficients automatically [13,15]. Volpe et al. proposed an automated framework supporting end-to-end conversion from classical optimization models to quantum solvers [84]. However, none of these tools performs constraint type analysis, penalty weight calibration, or hardware-aware bina-

rization precision control. The complete automation of the pipeline from a structured MILP to a deployable QUBO therefore remains an open problem.

Even if the QUBO construction process is fully automated, encoding a large-scale problem into a single QUBO matrix still faces a severe scalability problem. When the problem contains continuous variables, each variable requires a K-bit binary expansion, causing the dimension of the QUBO matrix to grow quadratically with the product of the variable count and the encoding precision. The experiments in Section 6.4 will show that on a moderately sized facility location instance, the monolithic QUBO approach fails to reduce the optimality gap to an acceptable range within a bounded time budget [2]. This result is consistent with the discussion in Section 2.5, where embedding overhead is shown to scale unfavorably with problem density and size. To address this, the chapter introduces Benders decomposition as a scalability strategy. This method partitions the original MILP by variable type into a binary master problem and a continuous subproblem. The master problem is encoded as a compact QUBO suitable for quantum annealers. The subproblem is solved as a linear program by a classical solver. The two components exchange information through iteratively generated Benders cuts until convergence. This division of labor enables large-scale instances that exceed the capacity of a single quantum device to be solved within a hybrid framework [2].

The framework proposed in this chapter consists of two core stages and an optional decomposition path. The first stage uses an LLM to parse a natural language problem description into a structured MILP model. The LLM is guided to identify five component categories: sets, parameters, decision variables with their types, the objective function, and constraints. Accurate classification of variable types at this stage is critical because it directly determines the binarization logic in the subsequent QUBO conversion. The second stage is the core contribution of this chapter: an LLM-driven QUBO transformation engine. This engine programmatically implements the QUBO conversion pipeline described in Section 2.3. Equality constraints are converted into penalty terms of the form $(LHS - RHS)^2$. Inequality constraints are first augmented with bit-width-minimized slack variables and then converted into equality penalties. Special-structure constraints, such as the pairwise exclusion constraint $x_i + x_j \leq 1$, are recognized and encoded as the more compact penalty term $P \cdot x_i x_j$. For problems that exceed the capacity of the target quantum device, the framework activates a Benders decomposition path. The binary master problem is sent to the QUBO transformation engine, while the continuous subproblem is handled

by a classical solver.

The main contributions of this chapter are as follows. First, it proposes an LLM-driven MILP-to-QUBO transformation engine that generates QUBO matrices with constraint type analysis, penalty weight assignment, and hardware-aware binarization precision control from a structured MILP, without requiring quantum computing expertise from the user [2]. Second, it integrates this transformation engine into a Benders decomposition framework in which the binary master problem is processed in QUBO form by a quantum annealer or a classical QUBO solver, while the continuous subproblem is solved by a classical solver. This integration is validated experimentally on large-scale problem instances. Third, it verifies the correctness of the automatically generated QUBO on nine classes of classical combinatorial optimization problems, and evaluates the solving efficiency and optimality gap of the hybrid Benders decomposition method against direct MILP solving with Gurobi across different problem scales.

The remainder of this chapter is organized as follows. Section 6.2 discusses the literature most relevant to this chapter, focusing on LLM-assisted optimization modeling and automated QUBO generation. Section 6.3 describes the overall framework architecture in detail, including LLM-driven problem structuring, the design of the QUBO transformation engine, and the integration of Benders decomposition. Section 6.4 reports experimental results, covering QUBO generation correctness verification, a demonstration of the scalability bottleneck of monolithic QUBO, and the performance evaluation of the hybrid Benders decomposition method. Section 6.5 summarizes the chapter.

6.2 Related Work

Chapter 2 provides a comprehensive literature foundation covering the methodological chain from classical mathematical programming through QUBO to hybrid execution paths. This section does not repeat that review. Instead, it focuses on two specific directions that are directly relevant to the core contribution of this chapter: research on using large language models to convert optimization problems from natural language or structured descriptions into mathematical models automatically, and tools and frameworks that convert mathematical models into QUBO matrices.

Research on converting natural language problem descriptions into structured optimization models has advanced rapidly in recent years. The NL4Opt competition

divided this task into named entity recognition and formula generation subtasks, establishing early benchmarks for the field [81, 85]. Subsequent work has developed along two directions. Among prompt-based methods, OptiMUS constructed a multi-agent LLM system that generates solvable MILP code through modular state representation, automated test generation, and confidence-guided iterative correction, substantially outperforming direct prompting baselines on the NLP4LP and NL4Opt datasets [82]. Among fine-tuning-based methods, ORLM proposed a semi-automated data synthesis pipeline for fine-tuning open-source models [83], and LLMOPT employed multi-instruction supervised fine-tuning to improve modeling accuracy across a broader set of problem types [28]. The output of all these systems is an MILP model that can be processed by classical solvers such as Gurobi. The subsequent conversion from MILP to QUBO falls outside their scope.

At the QUBO generation level, several tools provide varying degrees of automation. PyQUBO offers a Python interface that allows users to define constraints and objective functions symbolically, and the library compiles them into a QUBO matrix automatically [56]. Chapter 3 used PyQUBO to construct the QUBO for the metaverse data marketplace problem. However, PyQUBO still requires the user to perform constraint analysis, penalty structure selection, and coefficient calibration manually. Its automation remains at the syntactic level rather than the semantic level. AutoQUBO adopted a data-driven approach: it extracts the constant, linear, and quadratic coefficients of a QUBO by systematically sampling the all-zeros vector, one-hot vectors, and two-hot vectors [15]. AutoQUBO v2 further separates the QUBO matrices for the objective function and constraints and automatically estimates a penalty weight based on the objective matrix [13]. However, AutoQUBO handles continuous variables through direct binarization. For a facility location problem with N facilities and M customers, if each continuous assignment variable is encoded with K binary bits, the total variable count becomes $N + NMK$ and the QUBO matrix dimension becomes $(N + NMK)^2$. This quadratic growth constitutes a severe scalability barrier when M or K is large. Volpe et al. proposed an automated framework that allows users to describe problems through an interface resembling classical optimization practice, with the system handling QUBO conversion and solver selection automatically [84]. This framework covers the full pipeline from problem description to solver dispatch, but does not perform constraint type analysis or hardware-aware penalty weight calibration. In summary, existing tools provide automation at individual stages, such as syntactic compilation, coefficient extraction,

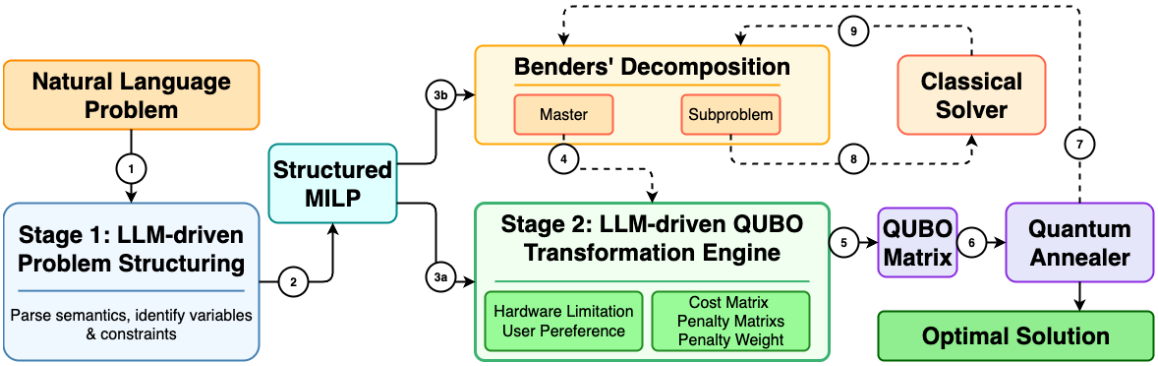


Figure 6.1: Overview of the automated QUBO formulation and hybrid orchestration framework

or solver dispatch, but the complete pipeline from a structured MILP to a QUBO with constraint analysis, penalty calibration, and binarization precision control has not been automated. The QUBO transformation engine proposed in this chapter fills this gap.

6.3 Framework Design

This section describes in detail the architecture of the LLM-driven framework proposed in this chapter. Section 6.3.1 presents the overall architecture and workflow. Section 6.3.2 describes the core design of the QUBO transformation engine. Section 6.3.3 explains the integration of Benders decomposition.

6.3.1 Overall Architecture and Workflow

The overall architecture of the framework is illustrated in Figure 6.1. It is an end-to-end pipeline that takes a natural language problem description as input.

The first stage of the pipeline is LLM-driven problem structuring. This stage uses an LLM to parse the unstructured problem description into a formal mathematical model. The LLM is guided to identify five component categories: index sets, parameter constants, decision variables with their types, the objective function with its optimization direction, and constraints. The LLM then synthesizes these components into a structured MILP that serves as the standardized input for subsequent transformation stages. The accurate classification of the variable types at this stage is critical because it directly determines the binarization logic in the QUBO conversion process. From the structured MILP, the framework supports two operational paths.

For smaller problems, the framework takes a direct conversion path, feeding MILP into the LLM-driven QUBO transformation engine in the second stage. For large-scale problems that exceed the capacity of current quantum hardware, the framework activates a hybrid decomposition path. In this path, the MILP is first partitioned by a Benders decomposition module into a combinatorially structured master problem and a linearly structured subproblem. The master problem is sent to the QUBO transformation engine, while the subproblem is handled by a classical solver. In the hybrid workflow, the QUBO solution of the master problem and the solution of the subproblem are used to generate Benders cuts, which are fed back to the decomposition module to refine the master problem. This iterative loop continues until the solution converges within a predefined optimality tolerance.

6.3.2 QUBO Transformation Engine

The second stage is the core of the framework: the LLM-driven QUBO transformation engine. This engine uses the LLM as a rule-based conversion executor through structured prompt engineering. The LLM is guided to systematically deconstruct the input structured MILP and resynthesize it into a Python class suitable for QUBO generation tools such as AutoQUBO [15]. This class explicitly separates the objective function from the constraint penalty terms.

The conversion executed by the LLM follows the QUBO transformation principles established in Section 2.3, with two programmatic enhancements. First, for standard constraint handling, the LLM automatically identifies the constraint type and applies the corresponding conversion rule. Equality constraints $LHS = RHS$ are converted into the penalty term $(LHS - RHS)^2$. Inequality constraints $LHS \leq RHS$ are converted into $(LHS + s - RHS)^2$ by introducing a slack variable s whose bit-width is minimized based on the constraint parameters. Second, the LLM is further guided to recognize special constraint structures that admit more compact QUBO representations. For example, for the pairwise exclusion constraint $x_i + x_j \leq 1$ over binary variables, the LLM recognizes that this constraint is violated only when $x_i = x_j = 1$ and encodes it directly as the penalty term $P \cdot x_i x_j$, avoiding the introduction of unnecessary slack variables.

For the binarization of non-binary variables, the framework adds hardware-aware precision control on top of the standard binary expansion described in Section 2.3. Specifically, if the target quantum annealer accepts a QUBO matrix of at most di-

mension D and the current problem already contains n_b native binary variables, the encoding bit-width k for each integer variable must satisfy $n_b + k \leq D$. The structured prompts instruct the LLM to compute and respect this hardware constraint, ensuring that the generated QUBO is physically solvable. For example, if a problem contains 15 binary variables and one integer variable with upper bound $U = 511$, and the target hardware limit is 24×24, the encoding precision for the integer variable cannot exceed $k = 9$ bits.

The final QUBO matrix consists of the cost matrix corresponding to the objective function and a weighted sum of all constraint penalty matrices:

$$\text{QUBO} = \text{Cost Matrix} + \sum_j P_j \cdot \text{Penalty Matrix}_j \tag{6.1}$$

where P_j is the penalty coefficient for the j-th constraint. The framework allows distinct penalty weights to be assigned to different constraints so as to reflect differences in their semantic priority. However, the experiments in Section 6.4 will show that the LLM can still produce incorrect penalty structures for certain problems. For this reason, automatically generated QUBOs are verified for correctness through classical solvers before deployment.

6.3.3 Hybrid Benders Decomposition

Although the QUBO transformation engine can process any suitably sized MILP, encoding a large-scale problem as a single QUBO typically produces a matrix that is too large and dense for effective solving. This section therefore integrates the transformation engine into a Benders decomposition framework.

The framework assumes that the source MILP can be expressed in the standard form of Equation 2.1 in Chapter 2, where $y \in \{0,1\}^p$ denotes the binary decision variables and $x \in \mathbb{R}^n_+$ denotes the continuous variables. Benders decomposition reformulates this problem into two smaller, linked subproblems. The master problem is an integer program involving only the binary variables y. It approximates the effect of the continuous variables through a set of Benders cuts that are added iteratively. The subproblem is a linear program that solves for the continuous variables x with the values of y fixed.

The integration point for the core contribution of this chapter lies in the master problem. Once isolated, the master problem is a pure binary optimization problem

and becomes the direct input for the QUBO transformation engine described in Section 6.3.2. The LLM converts the master problem into a compact, hardware-aware QUBO.

The hybrid workflow operates iteratively. In each iteration, the master-problem QUBO is solved to propose a new set of binary decisions. The subproblem is then solved with these decisions fixed. The subproblem solution is used to generate a new Benders cut, which is added to the master problem for the next iteration, progressively tightening the feasible solution space. In the experiments of this chapter, the master-problem QUBO is solved using a classical solver. This design verifies the correctness of the decomposition and conversion, establishes a performance baseline, and confirms the structural compatibility of the master-problem QUBO with quantum annealers, providing the foundation for subsequent deployment on physical quantum hardware.

6.4 Experimental Evaluation

This section reports two groups of experiments. The first evaluates the conversion correctness of the LLM-driven QUBO transformation engine on nine classes of classical combinatorial optimization problems. The second evaluates the solving performance and scalability of the hybrid Benders decomposition framework on capacitated facility location problem (CFLP) instances of varying size. Both groups of experiments start from an accurate structured MILP model. This premise is grounded in the fact that existing research has shown that LLMs can convert natural language descriptions into structured MILP models [28]. The QUBO transformation tasks were performed using the Qwen3-8B model deployed on a single NVIDIA A40 GPU. The hybrid Benders decomposition experiments were conducted on an Apple M4 Pro CPU using Gurobi 12.0.3, with test instances drawn from the OR-Library benchmark suite [86–88].

6.4.1 QUBO Conversion Correctness

The primary objective of the automated conversion framework is to produce QUBO models that are not only syntactically valid but also semantically equivalent to the source MILP. Conversion quality is assessed according to three criteria: (1) whether the output conforms to the standardized format required by tools such as AutoQUBO, with the cost function and constraint penalty terms explicitly separated; (2) whether the binarization strategies for non-binary decision variables and the introduction of

Problem	Struct.	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 6.1: Analysis of automated MILP-to-QUBO conversion correctness for nine classical optimization problems.

slack variables for inequality constraints are correct; and (3) whether the penalty function formulation for each constraint is mathematically correct [2].

Table 6.1 summarizes the conversion results for nine classes of classical optimization problems. Seven of the nine problems pass all three criteria. The two failures reveal current limitations of the LLM transformation engine.

The portfolio optimization problem contains only continuous decision variables. The LLM did not perform binarization, resulting in an incorrect penalty function. This result indicates that the reliability of the transformation engine decreases significantly when the problem contains no native binary variables.

The traveling salesman problem conversion successfully binarized the continuous auxiliary variables u_i from the Miller-Tucker-Zemlin subtour elimination constraints. However, for the inequality constraint $u_i - u_j + n \cdot x_{ij} \leq n - 1$, the penalty term gen-

erated by the LLM expands into a polynomial of degree higher than two after substituting the binarized representations, violating the quadratic requirement of QUBO. The correct approach is to introduce a binarized slack variable s_{ij} to convert the inequality into the equality $u_i - u_j + n \cdot x_{ij} + s_{ij} = n - 1$, then apply the standard quadratic penalty $(u_i - u_j + n \cdot x_{ij} + s_{ij} - (n-1))^2$.

6.4.2 Scalability of Hybrid Benders Decomposition

This subsection evaluates the solving performance and scalability of the hybrid Benders decomposition framework on CFLP instances. The experiments compare the proposed method against two baselines. Method 1 is direct MILP solving: Gurobi solves the complete CFLP MILP model and serves as the baseline for solution quality and classical performance. Method 2 is monolithic QUBO solving: the complete CFLP is converted into a single QUBO matrix using the framework and solved by Gurobi's quadratic programming capability, representing the performance of a non-decomposed QUBO approach. Method 3 is hybrid Benders decomposition (the proposed method): the CFLP is partitioned into a QUBO master problem and a linear programming subproblem according to the framework logic, with both solved by Gurobi to provide a fair comparison against Method 1.

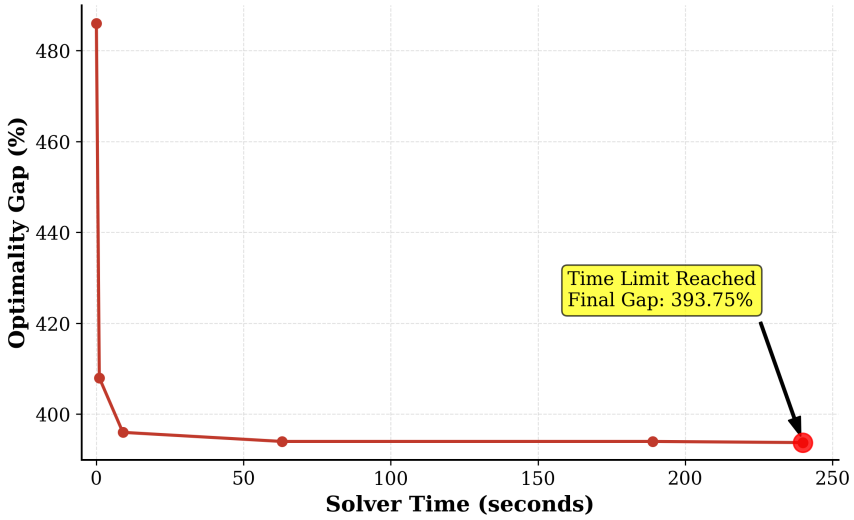


Figure 6.2: Convergence Failure of the Monolithic QUBO Approach on a 20 × 20 CFLP Instance.

Before evaluating the hybrid method, the scalability bottleneck of the monolithic QUBO approach is demonstrated. Method 2 is applied to a moderately sized CFLP

instance with 20 facilities and 20 customers [89]. Figure 6.2 shows the convergence trajectory of the optimality gap during solving. Despite rapid initial improvement, the solver stagnates quickly and does not reduce the gap below 393% within the 240-second time limit. This result quantifies the scalability barrier discussed in Section 6.1: the matrix size and density of the monolithic QUBO create an extremely difficult search landscape.

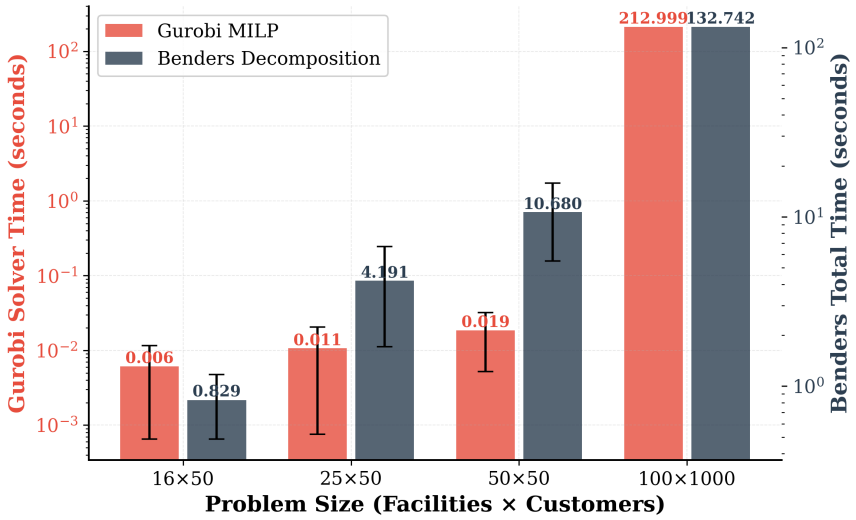


Figure 6.3: Scalability Comparison of the Hybrid Benders Decomposition against a Direct MILP Solver.

Figure 6.3 presents the performance comparison between hybrid Benders decomposition (Method 3) and direct Gurobi solving (Method 1) on CFLP instances scaling from 16×50 to 100×1000. For smaller instances (16×50 and 25×50), the fixed overhead of the Benders iterative loop makes it slower than the highly optimized direct Gurobi solver. However, the advantage of the decomposition approach becomes evident at the largest scale of 100 facilities and 1000 customers. The direct Gurobi solver required 213.0 seconds, while the hybrid framework obtained a solution with an optimality gap of only 0.2% in 132.7 seconds, reducing runtime by approximately 38%. These results were obtained with the master-problem QUBO solved by a classical solver. The QUBO structure of the master problem is designed to be compatible with quantum annealers, but whether a QPU backend can further improve performance at this problem scale depends on the embedding efficiency and sampling quality of the hardware. This will be investigated in future work.

Figures 6.4 and 6.5 provide a detailed view of the internal mechanics of the Benders decomposition process. Figure 6.4 shows the characteristic convergence pattern: the

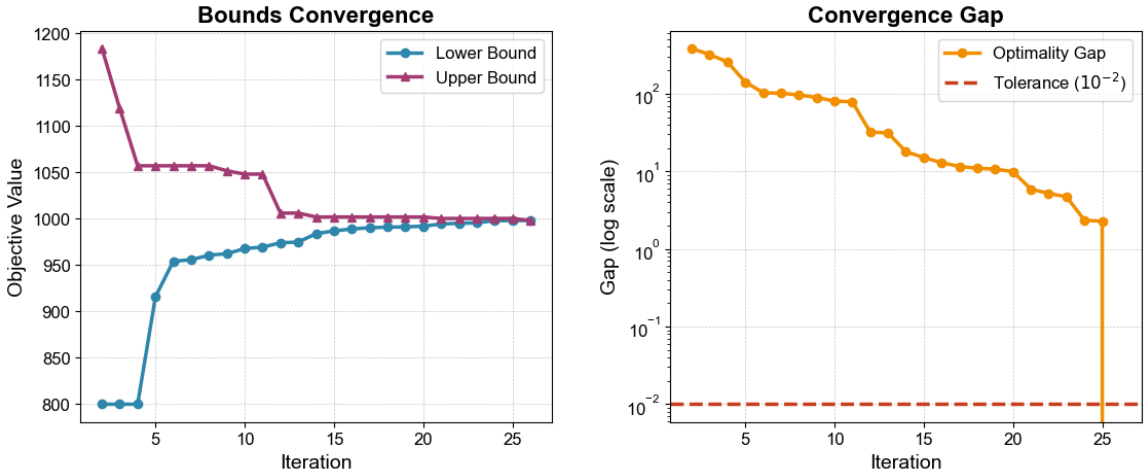


Figure 6.4: Overview of the convergence behavior in the CFLP Benders decomposition approach.

lower bound derived from the master problem increases monotonically while the upper bound from the subproblem decreases, closing the optimality gap robustly within 30 iterations. Figure 6.5 decomposes the objective value into three components, the master problem objective, the subproblem cost, and the total cost, illustrating the interaction dynamics between the combinatorial master problem and the linear subproblem as they guide the search toward convergence.

6.5 Summary

This chapter proposed an LLM-driven end-to-end framework that automates the conversion pipeline from a structured MILP to a hardware-aware QUBO and achieves scalable solving for large-scale problems through Benders decomposition. Experiments on nine classes of classical combinatorial optimization problems showed that the transformation engine correctly handles problems with diverse constraint types, but limitations remain for problems with purely continuous variables and constraint structures that produce higher-order terms. Scalability experiments on the capacitated facility location problem demonstrated that monolithic QUBO loses viability at moderate problem sizes, while hybrid Benders decomposition achieved a 38% runtime reduction relative to direct Gurobi solving on the largest test instance. This chapter addresses Research Gaps G1 and G4 simultaneously: the transformation engine eliminates manual dependence in QUBO modeling, and Benders decomposition separates the binary combinatorial core from the continuous subproblem so that prob-

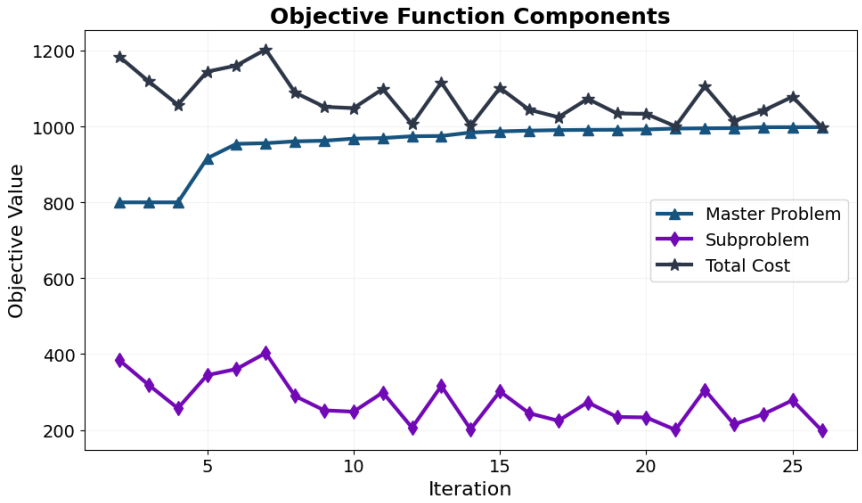


Figure 6.5: Evolution of Objective Function Components During Benders Decomposition.

lems exceeding the capacity of a single quantum device can be solved within a hybrid framework.

Chapter 7

Conclusions

This thesis has developed and validated a four-stage methodological progression for combinatorial decision optimization in networked systems, moving from classical modeling to QUBO reformulation, hybrid solution orchestration, and computational backend execution. In connection with the research questions introduced in Chapter 1 and the theoretical foundations established in Chapter 2, the central conclusion of this thesis is that, for the representative networked system domains investigated here, QUBO should not be understood as an isolated endpoint detached from problem semantics and execution conditions. Instead, it is more appropriately viewed as an intermediate abstraction layer that connects classical mathematical programming models with heterogeneous computational backends. From this standpoint, the thesis further shows that, under current limitations in quantum hardware scale, connectivity, and embedding overhead, the more realistic pathway is not to replace classical optimization entirely with quantum computation, but to construct a hybrid quantum-classical architecture with a clear division of labor, in which quantum or quantum-inspired modules are embedded into the broader solution process to strengthen the treatment of difficult discrete subproblems. Following this overall logic, the thesis addresses the research gaps identified in Chapter 1 from four perspectives: modelability, dynamic adaptability, decision expressiveness, and scalable orchestration. More specifically, QUBO modeling and validation in representative domains address the formulation dependence problem, the two-stage hybrid mechanism addresses the limitation of static optimization under dynamic conditions, the joint design of discrete and continuous decisions extends the practical scope of binary decision modeling, and automated QUBO generation together with decomposition-based hybrid solving advances the system-level feasibility of handling larger problem instances [3]. Overall, the con-

tribution of this thesis is not to establish QUBO as the universal final form for all network optimization problems, but to show that, within the networked system settings studied here, it can serve as a methodological bridge with cross-scenario reuse potential, more tightly linking problem formulation, hybrid solution architecture, and computational backend execution.

7.1 Limitations and Future Research Directions

To place the conclusions of Section 7.1 in proper perspective, they should be interpreted together with the scope and modeling assumptions under which the thesis operates. Although the thesis has validated an overall pathway from classical modeling to QUBO reformulation, hybrid solution procedures, and computational backend execution in representative networked systems, the proposed methodology still relies on several comparatively strong assumptions, including that the problem structure can be specified in advance, that key parameters can be stably encoded, and that the solution workflow can be organized within a unified optimization pipeline. Accordingly, this section does not restate the completed contributions of the thesis. Instead, it discusses four unresolved directions of broader methodological significance: moving beyond the assumption of pre-specified and stable problem formulations (Section 7.1.1), incorporating decision-aware learning into QUBO parameterization (Section 7.1.2), developing online and adaptive hybrid orchestration (Section 7.1.3), and establishing evaluation protocols that better reflect deployment-level realities (Section 7.1.4).

7.1.1 Beyond the centralized full-information assumption

A fundamental limitation of this thesis is not that all of its chapters remain within a purely static one-shot optimization paradigm, but rather that most of them still assume that the core problem representation can be specified in advance and remains relatively stable throughout the decision process. It should be acknowledged that the thesis has already taken partial but limited steps beyond this point. Chapter 4 adopts a two-stage mechanism that combines offline candidate topology generation with lightweight online selection, thereby avoiding global re-optimization at every time step [3]. Chapter 5 further separates global orchestration from node-level online scheduling, where the node-level scheduler acts only on locally observable states at discrete decision epochs and adaptively mixes a quantum-guided allocation with a

robust baseline through an exponential-weights update rule [23]. For this reason, the thesis should not be described simply as a fully centralized, fully informed, and purely one-shot optimization framework. A more precise characterization is that the present methodology still assumes that task dependencies, constraint graphs, admissible action spaces, and key cost parameters can be formulated as a unified optimization model before decision making begins, and that this representation does not change fundamentally during execution. Among these assumptions, uncertainty in cost parameters is comparatively more amenable to data-driven treatment, whereas changes in constraint structure or admissible actions alter the feasible region itself and therefore pose a deeper methodological challenge. In this sense, the thesis has already moved part of the way beyond static one-shot solving, but it has not yet entered the more difficult and realistic setting in which the decision maker must not only optimize under partial observations and time variation, but also continuously form, revise, or learn the problem representation itself. The following subsections discuss how this limitation may be progressively relaxed through learning-augmented orchestration, modeling-aware automation, and deployment-level evaluation.

7.1.2 Incorporating decision-aware learning into QUBO parameterization

Beyond the assumption of pre-specified and stable problem formulations, a more concrete future direction is to incorporate decision-aware learning into the parameterization process of QUBO models. More specifically, Elmachtoub and Grigas proposed the Smart Predict-then-Optimize framework together with the convex surrogate SPO+ loss, which directly trains prediction models against downstream decision error rather than parameter prediction error [90]. This direction does not require the SPO framework to be extended immediately to the final quadratic QUBO objective. Instead, because SPO+ already applies to polyhedral, convex, and mixed-integer optimization problems with linear objectives, the more natural entry point for this thesis lies in the upstream classical modeling layer before QUBO reformulation.

In the present setting, this means introducing such decision-aware training objectives into the MILP or two-stage stochastic integer programming models that serve as source formulations for QUBO conversion. The contextual information could then be used to learn cost coefficients, scenario-dependent parameters or quantities related to penalty construction, and these learned outputs could be propagated through the

existing MILP-to-QUBO transformation pipeline [91]. The significance of such a direction is that the learning component would no longer aim only at improving parameter fit but would instead be trained to support downstream combinatorial decision quality. A concrete example arises in the metaverse data marketplace problem of Chapter 3, where the cost structure in the stochastic integer programming model depends on demand scenario probabilities [1]. If those probabilities drift over time, the originally fixed QUBO coefficients may become progressively stale, which illustrates why static parameterization is unlikely to remain effective in dynamic data environments [1]. At the same time, the mapping from learned MILP parameters to the final QUBO representation is not transparent or linear. Binarization, penalty embedding, and slack-variable introduction jointly affect how upstream parameter perturbations propagate to final decision quality. As a result, end-to-end decision-aware training would require the reformulation chain itself to be differentiable, or at least compatible with the learning objective. If one goes further and seeks to define a decision-aware loss directly over the quadratic objective $x^\top Qx$, the theoretical challenge becomes substantially harder, because the current SPO+ surrogate relies on the linear structure of the objective in both the cost vector and the decision variable, and this property no longer holds in the QUBO setting. Recent work has begun to explore learning QUBO matrices directly from data, providing initial evidence that data-driven QUBO construction is feasible [92]. Overall, the more actionable near-term path is therefore to first establish a decision-aware learning interface between the upstream classical optimization layer and the QUBO reformulation layer, and to treat this as a first step toward data-driven QUBO parameterization.

7.1.3 Toward online and adaptive hybrid orchestration

Different from the previous subsection, which focuses mainly on parameter learning and updates to the problem representation, another future direction that follows more directly from the present thesis is to move hybrid quantum-classical architectures from predesigned static workflows toward online and adaptive orchestration. Although Chapters 4 and 5 already go part of the way beyond static one-shot solving through offline candidate generation with online selection and through the separation of global orchestration from node-level scheduling, their core orchestration logic is still largely fixed in advance by the designer [3]. This limitation involves two distinct levels of orchestration decisions.

The first level is intra-pipeline orchestration: within a single QUBO solving episode, deciding when to apply local repair, when to resample, and when to switch from quantum to classical postprocessing. Chapter 2 has already noted that even within the open-source dwave-hybrid framework, hybrid workflow performance is highly sensitive to annealing time, number of reads, embedding scheme, and workflow variants, and that designing a hybrid workflow that adapts across different problems and execution stages remains an unsolved engineering problem. The second level is inter-episode orchestration: across sequential decision epochs, deciding whether to continue with an existing QUBO instance, update the candidate solution set, switch solvers, or trigger problem reformulation. The exponential-weights (Hedge) mechanism in Chapter 5 provides an initial foundation for adaptive orchestration at this level [23]. That mechanism maintains an online trust weight between two competing experts, a quantum-guided allocator and a robust baseline allocator, and uses per-epoch surrogate loss as a learning signal to achieve adaptive mixing with an $O(\sqrt{T})$ cumulative regret guarantee [23]. A natural methodological extension is to scale this approach from two experts to multiple candidate solver modules, including quantum annealers, simulated annealing, classical exact solvers, local repair procedures, and QUBO regeneration, and from a fixed QUBO instance to an updatable problem representation [23].

Recent work on neural solver selection has suggests that different combinatorial optimization solvers exhibit complementary strengths across problem instances, and that a learned selection model can allocate each instance to the most suitable solver at the instance level [93]. Following this perspective, a natural extension of the present framework is to introduce online learning into the orchestration layer itself, so that the system can adaptively select among available orchestration actions based on past solving performance, current environmental disturbances, and available time budget. In this design, candidate learning signals include end-to-end decision quality, wall-clock time to feasibility, or energy landscape convergence rate. In this way, the learning objective of the hybrid architecture would evolve beyond the paradigm of first learning parameters and then solving, toward a tighter closed loop in which the system also learns how to orchestrate the interaction among quantum modules, classical solvers, and lightweight online control [91].

7.1.4 Toward deployment-level evaluation

Beyond algorithm design itself, another limitation of the present thesis is that its evaluation remains focused mainly on solution quality and per-run computational efficiency, without yet establishing a deployment-oriented end-to-end assessment framework. For QUBO-based and hybrid quantum-classical workflows, reporting only objective values, approximate optimality, or solver-internal runtime is insufficient to characterize the real decision cost of the system in practice. As noted in Chapter 2, QPU access time is an appropriate metric for solver benchmarking but is not sufficient for deployment-level latency evaluation. Future evaluation should therefore shift its target from an isolated solver to the end-to-end decision pipeline, systematically accounting for formulation time, reformulation overhead, embedding and sampling cost, feasibility recovery effort, solver-switching overhead, and control-plane response latency. Only at this level can different QUBO workflows, hybrid strategies, and computational backends be compared more objectively in terms of their actual suitability for dynamic networked systems. Future research should therefore pursue not only stronger solution methods, but also a deployment-level evaluation protocol that avoids mistaking local solver advantages for system-level gains.

7.2 Final Remarks

The contributions summarized in the conclusions presented at the beginning of this chapter and the limitations discussed in Section 7.1 together define the position of this thesis within the field. The methodological chain established in this thesis demonstrates that QUBO-driven hybrid quantum-classical optimization is no longer limited to single-problem proof of concept, but can play a systematic role across multiple stages from modeling to execution. At the same time, the limitations identified in this thesis are equally important: fixed problem representations, static orchestration logic, and solver-level evaluation are not yet sufficient to support reliable deployment in continuously evolving networked environments. Taken together, the final significance of this thesis lies not in providing a closed endpoint, but in offering a validated methodological pathway on which subsequent research can build by introducing data-driven parameterization, adaptive orchestration, and deployment-level assessment, thereby advancing the role of QUBO from a static modeling tool toward a component of closed-loop and evolvable decision infrastructure.

Appendix A

List of Abbreviations

C

CFLP: Capacitated facility location problem
CIM: Coherent Ising machine
CPU: Central processing unit
CU: Centralized unit

D

DU: Distributed unit

G

GPU: Graphics processing unit

I

ILP: Integer linear programming

L

LLM: Large language model

M

MILP: Mixed integer linear programming
MSP: Metaverse service provider

N

NP: Nondeterministic polynomial time
NUM: Network utility maximization

O

O-RAN: Open radio access network

Q

Q-GARS: Quantum Guided Adaptive Robust Scheduling
QA: Quantum annealing
QoS: Quality of service
QPU: Quantum processing unit
QUBO: Quadratic unconstrained binary optimization

R

RIC: RAN intelligent controller
RU: Radio unit

S

SA: Simulated annealing
SDN: Software defined networking
SINR: Signal to interference plus noise ratio
SIP: Stochastic integer programming
SMO: Service management and orchestration
SQA: Simulated quantum annealing
SRPT: Shortest remaining processing time

T

TGA: Target geographical area

U

UAV: Unmanned aerial vehicle

V

VNF: Virtual network function

Bibliography

[1] H. Zhang and M. Emu, "Coherent Optical Quantum Computing-Aided Resource Optimization for Transportation Digital Twin Construction," in *2025 IEEE International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*, Nov. 2025, pp. 561–566. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/11344304

[2] H. Zhang, M. Emu, and S. Choudhury, "LLM-QUBO: An End-to-End Framework for Automated QUBO Transformation from Natural Language Problem Descriptions," *Proceedings of the AAAI Symposium Series*, vol. 7, no. 1, pp. 411–418, Nov. 2025. [Online]. Available: https://ojs.aaai.org/index.php/AAAI-SS/article/view/36913

[3] H. Zhang, M. Emu, and O. A. Dobre, "Quantum Takes Flight: Two-Stage Resilient Topology Optimization for UAV Networks," Feb. 2026, arXiv:2601.19724 [cs]. [Online]. Available: http://arxiv.org/abs/2601.19724

[4] M. Emu, S. Choudhury, and K. Salomaa, "Stochastic Resource Optimization for Metaverse Data Marketplace by Leveraging Quantum Neural Networks," *IEEE Transactions on Network and Service Management*, vol. 21, no. 3, pp. 2613–2623, Jun. 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10500866

[5] K. Wu, K.-W. Chin, and S. Soh, "Topology Construction for Max-Min Rate Optimization in Heterogeneous AAVs Networks," *IEEE Internet of Things Journal*, vol. 12, no. 12, pp. 20 203–20 214, Jun. 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10891559

[6] K. Fu, W. Zhang, Q. Chen, D. Zeng, and M. Guo, "Adaptive Resource Efficient Microservice Deployment in Cloud-Edge Continuum," *IEEE Transactions on*

Parallel and Distributed Systems, vol. 33, no. 08, pp. 1825–1840, Aug. 2022. [Online]. Available: https://www.computer.org/csdl/journal/td/2022/08/09615028/1yyhxiLRpAs

[7] G. Codato and M. Fischetti, "Combinatorial Benders' Cuts for Mixed-Integer Linear Programming," *Operations Research*, vol. 54, no. 4, pp. 756–766, Aug. 2006. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/opre.1060.0286

[8] H. Liu and I. Sabek, "Towards a Hybrid Quantum-Classical Computing Framework for Database Optimization Problems in Real Time Setup," Feb. 2026, arXiv:2602.14263 [cs]. [Online]. Available: http://arxiv.org/abs/2602.14263

[9] T. Lamza, J. Zawalska, K. Jurek, M. Sterzel, and K. Rycerz, "QHyper: an integration library for hybrid quantum-classical optimization," *CoRR*, vol. abs/2409.15926, 2024, arXiv:2409.15926 [cs]. [Online]. Available: https://doi.org/10.48550/arXiv.2409.15926

[10] S. Kim, S.-W. Ahn, I.-S. Suh, A. W. Dowling, E. Lee, and T. Luo, "Quantum annealing for combinatorial optimization: a benchmarking study," *npj Quantum Information*, vol. 11, no. 1, p. 77, May 2025. [Online]. Available: https://www.nature.com/articles/s41534-025-01020-1

[11] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, Feb. 2014. [Online]. Available: https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2014.00005/full

[12] F. W. Glover, G. A. Kochenberger, R. Hennig, and Y. Du, "Quantum bridge analytics I: a tutorial on formulating and using QUBO models," *Ann. Oper. Res.*, vol. 314, no. 1, pp. 141–183, 2022. [Online]. Available: https://doi.org/10.1007/s10479-022-04634-2

[13] J. Pauckert, M. Ayodele, M. García, S. Georgescu, and M. Parizy, "AutoQUBO v2: Towards Efficient and Effective QUBO Formulations for Ising Machines," *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pp. 227–230, 2023. [Online]. Available: https://doi.org/10.1145/3583133.3590662

[14] F. Glover, G. Kochenberger, M. Ma, and Y. Du, "Quantum Bridge Analytics II: QUBO-Plus, network optimization and combinatorial chaining for asset exchange," *Annals of Operations Research*, vol. 314, no. 1, pp. 185 – 212, 2022. [Online]. Available: https://doi.org/10.1007/s10479-022-04695-3

[15] A. Moraglio, S. Georgescu, and P. Sadowski, "AutoQubo: data-driven automatic QUBO generation," *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2232–2239, 2022. [Online]. Available: https://doi.org/10.1145/3520304.3533965

[16] A. Abbas, A. Ambainis, B. Augustino, A. Bärtschi, H. Buhrman, C. Coffrin, G. Cortiana, V. Dunjko, D. J. Egger, B. G. Elmegreen, N. Franco, F. Fratini, B. Fuller, J. Gacon, C. Gonciulea, S. Gribling, S. Gupta, S. Hadfield, R. Heese, G. Kircher, T. Kleinert, T. Koch, G. Korpas, S. Lenk, J. Marecek, V. Markov, G. Mazzola, S. Mensa, N. Mohseni, G. Nannicini, C. O'Meara, E. P. Tapia, S. Pokutta, M. Proissl, P. Rebentrost, E. Sahin, B. C. B. Symons, S. Tornow, V. Valls, S. Woerner, M. L. Wolf-Bauwens, J. Yard, S. Yarkoni, D. Zechiel, S. Zhuk, and C. Zoufal, "Challenges and opportunities in quantum optimization," *Nature Reviews Physics*, vol. 6, no. 12, pp. 718–735, Dec. 2024. [Online]. Available: https://www.nature.com/articles/s42254-024-00770-9

[17] D-Wave Quantum Inc., "Reformulating a Problem." [Online]. Available: https://docs.dwavequantum.com/en/latest/quantum_research/reformulating.html#qpu-reformulating

[18] Z. Zhao, L. Fan, and Z. Han, "Hybrid Quantum Benders' Decomposition For Mixed-integer Linear Programming," *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 2536–2540, Apr. 2022, conference Name: 2022 IEEE Wireless Communications and Networking Conference (WCNC) ISBN: 9781665442664 Place: Austin, TX, USA. [Online]. Available: https://ieeexplore.ieee.org/document/9771632/

[19] Z. Peng, D. D. Roux, Carnegie Mellon University, D. E. B. Neira, and Purdue University, "Hybrid Quantum Branch-and-Bound Method for Quadratic Unconstrained Binary Optimization," 2025. [Online]. Available: http://wscg.zcu.cz/QC-2025/papers-CSRN/A67.pdf

[20] J. Zhang, G. Lo Bianco, and J. C. Beck, "Solving Job-Shop Scheduling Problems with QUBO-Based Specialized Hardware," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 32, pp. 404–412, Jun. 2022. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/19826

[21] R. A. Quintero and L. F. Zuluaga, "QUBO Formulations of Combinatorial Optimization Problems for Quantum Computing Devices," in *Encyclopedia of Optimization*. Springer, Cham, 2023, pp. 1–13. [Online]. Available: https://link.springer.com/rwe/10.1007/978-3-030-54621-2_853-1

[22] K. Suda, S. Naito, and Y. Hasegawa, "Sparse QUBO Formulation for Efficient Embedding via Network-Based Decomposition of Equality and Inequality Constraints," Jan. 2026, arXiv:2601.18108 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2601.18108

[23] H. Zhang and M. Emu, "Q-GARS: Quantum-inspired Robust Microservice Chaining Scheduling," Mar. 2026, arXiv:2603.23127 [cs]. [Online]. Available: http://arxiv.org/abs/2603.23127

[24] QBoson Inc., "Kaiwu SDK for development and research on coherent ising machine." [Online]. Available: https://www.qboson.com/

[25] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual." [Online]. Available: https://www.gurobi.com

[26] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983. [Online]. Available: https://www.science.org/doi/10.1126/science.220.4598.671

[27] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, Mar. 1998. [Online]. Available: https://doi.org/10.1057/palgrave.jors.2600523

[28] C. Jiang, X. Shu, H. Qian, X. Lu, J. Zhou, A. Zhou, and Y. Yu, "LLMOPT: Learning to Define and Solve General Optimization Problems from Scratch," in *Proceedings of the Thirteenth International Conference on Learning*

Representations, vol. abs/2410.13213. OpenReview.net, 2025, arXiv:2410.13213 [cs]. [Online]. Available: https://openreview.net/forum?id=9OMvtboTJg

[29] M. Min and A. Chinchuluun, "Optimization in Wireless Networks," in *Handbook of Optimization in Telecommunications*, M. G. C. Resende and P. M. Pardalos, Eds. Boston, MA: Springer US, 2006, pp. 891–915. [Online]. Available: https://doi.org/10.1007/978-0-387-30165-5_31

[30] M. Emu, S. Choudhury, and K. Salomaa, "Resource Optimization of SFC Embedding for IoT Networks Using Quantum Computing," in *2022 IEEE 27th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Nov. 2022, pp. 83–88, iSSN: 2378-4873. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9966892

[31] H. Du, J. Liu, D. Niyato, J. Kang, Z. Xiong, J. Zhang, and D. I. Kim, "Attention-Aware Resource Allocation and QoE Analysis for Metaverse xURLLC Services," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 7, pp. 2158–2175, Jul. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10144339

[32] N. H. Chu, D. T. Hoang, D. N. Nguyen, K. T. Phan, E. Dutkiewicz, D. Niyato, and T. Shu, "MetaSlicing: A Novel Resource Allocation Framework for Metaverse," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4145–4162, May 2024. [Online]. Available: https://doi.org/10.1109/TMC.2023.3288085

[33] H. Hu, X. Zhu, F. Zhou, W. Wu, R. Qingyang Hu, and H. Zhu, "Resource Allocation for Multi-Modal Semantic Communication in UAV Collaborative Networks," *IEEE Transactions on Communications*, vol. 73, no. 9, pp. 7599–7616, Sep. 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10930657

[34] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware VNF Chain Deployment with Efficient Resource Reuse at Network Edge," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Jul. 2020, pp. 267–276, iSSN: 2641-9874. [Online]. Available: https://ieeexplore.ieee.org/document/9155345

[35] Y.-F. Liu, T.-H. Chang, M. Hong, Z. Wu, A. Man-Cho So, E. A. Jorswieck, and W. Yu, "A Survey of Recent Advances in Optimization Methods for Wireless Communications," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 11, pp. 2992–3031, Nov. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10636212

[36] Y. Mao, X. Shang, and Y. Yang, "Joint Resource Management and Flow Scheduling for SFC Deployment in Hybrid Edge-and-Cloud Network," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, May 2022, pp. 170–179, iSSN: 2641-9874. [Online]. Available: https://ieeexplore.ieee.org/document/9796884

[37] J. Wang, M. Dong, B. Liang, G. Boudreau, and H. Abou-Zeid, "Delay-Tolerant OCO With Long-Term Constraints: Algorithm and Its Application to Network Resource Allocation," *IEEE/ACM Transactions on Networking*, vol. 31, no. 1, pp. 147–163, Feb. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9831158

[38] G. Nemhauser and L. Wolsey, "Linear Programming," in *Integer and Combinatorial Optimization*. John Wiley & Sons, Ltd, 1988, pp. 27–49, section: I.2 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118627372.ch2. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118627372.ch2

[39] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*, ser. Springer Series in Operations Research and Financial Engineering. New York, NY: Springer, 2011. [Online]. Available: https://link.springer.com/10.1007/978-1-4614-0237-4

[40] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski, *Robust Optimization*. Princeton University Press, Aug. 2009, section: Robust Optimization. [Online]. Available: https://doi.org/10.1515/9781400831050.3

[41] P. M. Pardalos and G. P. Rodgers, "Computational aspects of a branch and bound algorithm for quadratic zero-one programming," *Computing*, vol. 45, no. 2, pp. 131–144, Jun. 1990. [Online]. Available: https://doi.org/10.1007/BF02247879

[42] A. Verma and M. Lewis, "Penalty and partitioning techniques to improve performance of QUBO solvers," *Discrete Optimization*, vol. 44, p. 100594, May

2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1572528620300281

[43] M. Ayodele, "Penalty Weights in QUBO Formulations: Permutation Problems," L. Pérez Cáceres and S. Verel, Eds., vol. 13222. Cham: Springer International Publishing, 2022, pp. 159–174, book Title: Evolutionary Computation in Combinatorial Optimization Series Title: Lecture Notes in Computer Science. [Online]. Available: https://link.springer.com/10.1007/978-3-031-04148-8_11

[44] C. Coffrin and M. Vuffray, "Quantum Annealing," in *Encyclopedia of Optimization*, P. M. Pardalos and O. A. Prokopyev, Eds. Cham: Springer International Publishing, 2024, pp. 1–8. [Online]. Available: https://link.springer.com/10.1007/978-3-030-54621-2_855-1

[45] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, "Evidence for quantum annealing with more than one hundred qubits," *Nature Physics*, vol. 10, no. 3, pp. 218–224, Mar. 2014. [Online]. Available: https://www.nature.com/articles/nphys2900

[46] P. Date, R. Patton, C. Schuman, and T. Potok, "Efficiently embedding QUBO problems on adiabatic quantum computers," *Quantum Information Processing*, vol. 18, no. 4, p. 117, Mar. 2019. [Online]. Available: https://doi.org/10.1007/s11128-019-2236-3

[47] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, "The unconstrained binary quadratic programming problem: a survey," *Journal of Combinatorial Optimization*, vol. 28, no. 1, pp. 58–81, Jul. 2014. [Online]. Available: https://doi.org/10.1007/s10878-014-9734-0

[48] M. Mohseni, M. M. Rams, S. V. Isakov, D. Eppens, S. Pielawa, J. Strumpfer, S. Boixo, and H. Neven, "Sampling diverse near-optimal solutions via algorithmic quantum annealing," *Physical Review E*, vol. 108, no. 6, p. 065303, Dec. 2023. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.108.065303

[49] A. Zucca, H. Sadeghi, M. Mohseni, and M. H. Amin, "Diversity metric for evaluation of quantum annealing," Oct. 2021, arXiv:2110.10196 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2110.10196

[50] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, "Quantum annealing for industry applications: introduction and review," *Reports on Progress in Physics*, vol. 85, no. 10, p. 104001, Sep. 2022. [Online]. Available: https://doi.org/10.1088/1361-6633/ac8c54

[51] M. Emu, S. Choudhury, and K. Salomaa, "Warm and Cold Start Quantum Annealing for Metaverse Resource Optimization," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 1057–1071, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10415636

[52] J. Raymond, R. Stevanovic, W. Bernoudy, K. Boothby, C. C. McGeoch, A. J. Berkley, P. Farré, J. Pasvolsky, and A. D. King, "Hybrid Quantum Annealing for Larger-than-QPU Lattice-structured Problems," *ACM Transactions on Quantum Computing*, vol. 4, no. 3, pp. 17:1–17:30, Apr. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3579368

[53] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, p. 015002, Jan. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.90.015002

[54] K. Boothby, P. Bunyk, J. Raymond, and A. Roy, "Next-Generation Topology of D-Wave Quantum Processors," Feb. 2020, arXiv:2003.00133 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2003.00133

[55] M. Emu, S. Choudhury, and K. Salomaa, "Quantum Neural Networks driven Stochastic Resource Optimization for Metaverse Data Marketplace," in *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, Jun. 2023, pp. 242–246, iSSN: 2693-9789. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10175433

[56] M. Zaman, K. Tanahashi, and S. Tanaka, "PyQUBO: Python Library for Mapping Combinatorial Optimization Problems to QUBO Form," *IEEE Transactions on Computers*, vol. 71, no. 4, pp. 838–850, Apr. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9369010

[57] S. Zhang, W. Y. B. Lim, W. C. Ng, Z. Xiong, D. Niyato, X. S. Shen, and C. Miao, "Toward Green Metaverse Networking: Technologies, Advancements, and Future Directions," *IEEE Network*, vol. 37, no. 5, pp. 223–232, Sep. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10012292

[58] N. Torkzaban, A. Gholami, and J. S. Baras, "DeFeL-FUN: Decentralized Federated Learning Framework for UAV-Enabled Networks," in *ICC 2025 - IEEE International Conference on Communications*, Jun. 2025, pp. 6554–6559. [Online]. Available: https://ieeexplore.ieee.org/document/11161012

[59] S. Garg, A. Ihler, E. S. Bentley, and S. Kumar, "A Cross-Layer, Mobility, and Congestion-Aware Routing Protocol for UAV Networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 4, pp. 3778–3796, Aug. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9999549

[60] A. Chapnevis and E. Bulut, "UAV Mesh Network Trajectory Planning for Age Optimal Data Collection in Infrastructureless Areas," in *ICC 2024 - IEEE International Conference on Communications*, Jun. 2024, pp. 1563–1568. [Online]. Available: https://ieeexplore.ieee.org/document/10622459

[61] A. A. Baktayan, A. T. Zahary, A. Sikora, and D. Welte, "Computational offloading into UAV swarm networks: a systematic literature review," *EURASIP Journal on Wireless Communications and Networking*, vol. 2024, no. 1, p. 69, Sep. 2024. [Online]. Available: https://doi.org/10.1186/s13638-024-02401-4

[62] Y. Lin, C. Xu, and C. Wang, "Multi-Objective Routing Optimization Using Coherent Ising Machine in Wireless Multihop Networks," in *ICC 2025 - IEEE International Conference on Communications*, Jun. 2025, pp. 2370–2375. [Online]. Available: https://ieeexplore.ieee.org/document/11162085

[63] S.-G. Jeong, P. D. A. Duc, Q. V. Do, D.-I. Noh, N. X. Tung, T. V. Chien, Q.-V. Pham, M. Hasegawa, H. Sekiya, and W.-J. Hwang, "Quantum-Annealing-Based Sum Rate Maximization for Multi-UAV-Aided Wireless Networks," *IEEE Internet of Things Journal*, vol. 12, no. 12, pp. 21 225–21 239, Jun. 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10907925/

[64] G. Xiong, J. Guo, K. Parsons, Y. Nagai, T. Sumi, P. Orlik, and J. Li, "UAV Aided Smart Agriculture Networks: A Multi-Agent Reinforcement Learning Approach," in *ICC 2025 - IEEE International Conference on Communications*, Jun. 2025, pp. 3075–3081. [Online]. Available: https://ieeexplore.ieee.org/document/11161267

[65] M. Emu, S. Choudhury, and K. Salomaa, "Quantum Computing Empowered Metaverse: An Approach for Resource Optimization," in *ICC 2023 - IEEE*

International Conference on Communications, May 2023, pp. 2412–2418. [Online]. Available: https://ieeexplore.ieee.org/document/10278858

[66] M. Yan, L. Zhang, W. Jiang, C. A. Chan, A. F. Gygax, and A. Nirmalathas, "Energy Consumption Modeling and Optimization of UAV-Assisted MEC Networks Using Deep Reinforcement Learning," *IEEE Sensors Journal*, vol. 24, no. 8, pp. 13 629–13 639, Apr. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10458908

[67] Y. Hu, H. Wang, L. Wang, M. Hu, K. Peng, and B. Veeravalli, "Joint Deployment and Request Routing for Microservice Call Graphs in Data Centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 11, pp. 2994–3011, Nov. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10247200

[68] J. Li, N. K. Sharma, D. R. K. Ports, and S. D. Gribble, "Tales of the Tail: Hardware, OS, and Application-level Sources of Tail Latency," in *Proceedings of the ACM Symposium on Cloud Computing*, ser. SOCC '14. New York, NY, USA: Association for Computing Machinery, Nov. 2014, pp. 1–14. [Online]. Available: https://dl.acm.org/doi/10.1145/2670979.2670988

[69] M. Emu, S. Choudhury, and K. Salomaa, "Robust Virtual Network Function Optimization Under Post-Failure Uncertainty: Classical & Quantum Computing," in *2024 20th International Conference on the Design of Reliable Communication Networks (DRCN)*, May 2024, pp. 129–135, iSSN: 2766-3647. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10539145

[70] X. Zhu, G. She, B. Xue, Y. Zhang, Y. Zhang, X. K. Zou, X. Duan, P. He, A. Krishnamurthy, M. Lentz, D. Zhuo, and R. Mahajan, "Dissecting Overheads of Service Mesh Sidecars," in *Proceedings of the 2023 ACM Symposium on Cloud Computing*, ser. SoCC '23. New York, NY, USA: Association for Computing Machinery, Oct. 2023, pp. 142–157. [Online]. Available: https://dl.acm.org/doi/10.1145/3620678.3624652

[71] M. Purohit, Z. Svitkina, and R. Kumar, "Improving Online Algorithms via {ML} Predictions," in *Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, vol. 31. Curran Associates, Inc., 2018, pp.

9684–9693. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html

[72] A. Lindermayr and N. Megow, "Permutation Predictions for Non-Clairvoyant Scheduling," *ACM Trans. Parallel Comput.*, vol. 12, no. 2, pp. 4:1–4:26, May 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3711872

[73] D. Harutyunyan, N. Shahriar, R. Boutaba, and R. Riggio, "Latency-Aware Service Function Chain Placement in 5G Mobile Networks," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, Jun. 2019, pp. 133–141. [Online]. Available: https://ieeexplore.ieee.org/document/8806646

[74] N. Kiji, T. Sato, R. Shinkuma, and E. Oki, "Virtual Network Function Placement and Routing Model for Multicast Service Chaining Based on Merging Multiple Service Paths," in *2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR)*, May 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8807998

[75] A. Kawabata, B. C. Chatterjee, and E. Oki, "MHND: Multi-Homing Network Design Model for Delay Sensitive Distributed Processing Applications," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, Jan. 2023, pp. 348–353, iSSN: 2331-9860. [Online]. Available: https://ieeexplore.ieee.org/document/10059846

[76] M. Inoue and E. Oki, "Efficient Approximation Algorithms for Server Allocation to Minimize Total Capacity under Server Failure," *IEEE Transactions on Computers*, vol. 75, no. 1, pp. 335–349, Jan. 2026. [Online]. Available: https://ieeexplore.ieee.org/document/11224627/

[77] L. F. Pérez Armas, S. Creemers, and S. Deleplanque, "Solving the resource constrained project scheduling problem with quantum annealing," *Scientific Reports*, vol. 14, no. 1, p. 16784, Jul. 2024. [Online]. Available: https://www.nature.com/articles/s41598-024-67168-6

[78] J. Fried, Z. Ruan, A. Ousterhout, and A. Belay, "Caladan: mitigating interference at microsecond timescales," in *Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'20. USA: USENIX Association, Nov. 2020, pp. 281–297. [Online]. Available: https://dl.acm.org/doi/10.5555/3488766.3488782

[79] D. P. Palomar and M. Chiang, "Alternative Distributed Algorithms for Network Utility Maximization: Framework and Applications," *IEEE Transactions on Automatic Control*, vol. 52, no. 12, pp. 2254–2269, Dec. 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4395184

[80] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge ; New York: Cambridge University Press, 2006.

[81] R. Ramamonjison, T. T. L. Yu, R. Li, H. Li, G. Carenini, B. Ghaddar, S. He, M. Mostajabdaveh, A. Banitalebi{-}Dehkordi, Z. Zhou, and Y. Zhang, "NL4Opt Competition: Formulating Optimization Problems Based on Their Natural Language Descriptions," in *Proceedings of the NeurIPS 2022 Competition Track*, ser. Proceedings of Machine Learning Research, vol. 220. PMLR, 2021, pp. 189–203, arXiv:2303.08233 [cs]. [Online]. Available: https://proceedings.mlr.press/v220/ramamonjison22a.html

[82] A. AhmadiTeshnizi, W. Gao, and M. Udell, "OptiMUS: scalable optimization modeling with (MI)LP solvers and large language models," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24, vol. 235. Vienna, Austria: JMLR.org, Jul. 2024, pp. 577–596.

[83] C. Huang, Z. Tang, S. Hu, R. Jiang, X. Zheng, D. Ge, B. Wang, and Z. Wang, "ORLM: A Customizable Framework in Training Large Models for Automated Optimization Modeling," *Operations Research*, May 2025. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/opre.2024.1233

[84] D. Volpe, N. Quetschlich, M. Graziano, G. Turvani, and R. Wille, "Towards an Automatic Framework for Solving Optimization Problems with Quantum Computers," in *2024 IEEE International Conference on Quantum Software (QSW)*, Jul. 2024, pp. 46–57. [Online]. Available: https://ieeexplore.ieee.org/document/10646509

[85] R. Ramamonjison, H. Li, T. T. Yu, S. He, V. Rengan, A. Banitalebi-Dehkordi, Z. Zhou, and Y. Zhang, "Augmenting Operations Research with Auto-Formulation of Optimization Models from Problem Descriptions," 2022, version Number: 2. [Online]. Available: https://arxiv.org/abs/2209.15565

[86] R. K. Ahuja, J. B. Orlin, S. Pallottino, M. P. Scaparra, and M. G. Scutellà, "A Multi-Exchange Heuristic for the Single-Source Capacitated Facility Location

Problem," *Management Science*, vol. 50, no. 6, pp. 749–760, Jun. 2004. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/mnsc.1030.0193

[87] C.-H. Chen and C. Ting, "Combining Lagrangian heuristic and Ant Colony System to solve the Single Source Capacitated Facility Location Problem," *Transportation Research Part E-logistics and Transportation Review*, vol. 44, no. 6, pp. 1099–1122, 2008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1366554507000890

[88] G. Guastaroba and M. Speranza, "A heuristic for BILP problems: The Single Source Capacitated Facility Location Problem," *Eur. J. Oper. Res.*, vol. 238, no. 2, pp. 438–450, 2014. [Online]. Available: https://doi.org/10.1016/j.ejor.2014.04.007

[89] V. Beresnev, Y. Kochetov, M. Pashchenko, E. Goncharov, and A. Plyasunov, "Discrete Location Problems Benchmark library," 2011. [Online]. Available: http://old.math.nsc.ru/AP/benchmarks/english.html

[90] A. N. Elmachtoub and P. Grigas, "Smart "Predict, then Optimize"," *Management Science*, vol. 68, no. 1, pp. 9–26, Jan. 2022. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/mnsc.2020.3922

[91] M. Emu, T. Rahman, S. Choudhury, and K. Salomaa, "Next-Generation Vehicle Platooning: Leveraging Quantum Long Short-Term Memory Networks," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 02, Sep. 2024, pp. 585–586. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10820991

[92] J. Nüßlein, S. Zielinski, and C. Linnhoff-Popien, "Learning QUBO Formulations from Data," in *Innovations for Community Services*, S. Zielinski, G. Eichler, C. Erfurth, and G. Fahrnberger, Eds. Cham: Springer Nature Switzerland, 2025, pp. 209–227.

[93] C. Gao, H. Shang, K. Xue, and C. Qian, "Neural Solver Selection for Combinatorial Optimization," in *Proceedings of the 42nd International Conference on Machine Learning*. PMLR, Oct. 2025, pp. 18 528–18 549. [Online]. Available: https://proceedings.mlr.press/v267/gao25l.html