

Quantum Approaches to the 0/1 Multi-Knapsack Problem: QUBO Formulation, Penalty Parameter Characterization and Analysis

Evren Guney¹^a, Joachim Ehrental²^b and Thomas Hanne²^c

¹*Department of Industrial Engineering, MEF University, Maslak, Istanbul, Turkey*

²*School of Business, University of Applied Sciences and Arts Northwestern Switzerland FHNW, Brugg, Switzerland*

Keywords: Multi-Knapsack Problem, Quadratic Unconstrained Binary Optimization, Gate-Based Quantum Computing, Quantum Annealing, Quantum Simulation, Quantum Approximate Optimization Algorithm.

Abstract: The 0/1 Multi-Knapsack Problem (MKP) is a combinatorial optimization problem with applications in logistics, finance, and resource management. Advances in quantum computing have enabled the exploration of problems like the 0/1 MKP through Quadratic Unconstrained Binary Optimization (QUBO) formulations. This work develops QUBO formulations for the 0/1 MKP, with a focus on optimizing penalty parameters for encoding constraints. Using simulation experiments across quantum platforms, we evaluate the feasibility of solving small-scale instances of the 0/1 MKP. The results provide insights into the challenges and opportunities associated with applying quantum optimization methods for constrained resource allocation problems.

1 INTRODUCTION

The 0/1 Multi-Knapsack Problem (MKP) extends the classical knapsack problem to involve multiple knapsacks, each with its own capacity constraints. In this problem, a set of items is available, each characterized by a profit and weight. The objective is to determine how to assign these items to knapsacks such that the total profit is maximized while ensuring that the total weight in each knapsack does not exceed its capacity. Additionally, each item can be placed in at most one knapsack.


The 0/1 MKP is a well-known NP-hard problem, presenting computational challenges as the number of items and knapsacks increases. Classical optimization methods, such as exact algorithms, heuristics, and meta-heuristics, have been effective for solving small- to medium-scale instances. However, larger problem instances, particularly those with more complex constraints, require alternative approaches to address scalability and computational limitations—despite the recent growth in compute power.


Quantum computing offers an alternative to classical optimization by using quantum principles such


as superposition and entanglement to explore solution spaces more efficiently. Advances in quantum hardware and algorithms have enabled early investigations into solving combinatorial optimization problems on quantum simulators and devices. These devices predominantly operate on Quadratic Unconstrained Binary Optimization (QUBO) models, making them well-suited for initial research into problems such as 0/1 MKP alongside classical optimization approaches (Lucas (2014); Glover et al. (2022)).

A fundamental distinction between classical and quantum methods, such as those using QUBO, lies in their handling of constraints. Classical methods typically model constraints explicitly, ensuring feasibility through structured formulations. In contrast, QUBO models incorporate constraints directly into the objective function via penalty parameters. The effectiveness of solving the 0/1 MKP on quantum devices depends on the calibration of these penalty parameters, as they influence both the feasibility of solutions and algorithmic performance. These parameters may need to be adjusted according to the quantum platform or specific characteristics of the device, introducing additional complexity (Glover et al. (2019); Pecyna and Różycki (2024)).

Today's dominant platforms are annealing and gate-based systems. Quantum annealers (e.g., D-Wave) natively support QUBO formulations and offer a direct mapping of optimization problems (Pusey-

^a <https://orcid.org/0000-0001-7572-8627>

^b <https://orcid.org/0000-0003-2195-1465>

^c <https://orcid.org/0000-0002-5636-1660>

Nazzaro and Date (2020)), while gate-based quantum computers (e.g., IBM, Google, Quantinuum, IonQ) implement them through parameterized circuits (Arute et al. (2019)). These platforms differ in how they handle QUBO constraints and penalty parameters. By examining these differences experimentally, this work aims to provide practical insights into the current applicability and limitations of quantum platforms and their role alongside classical optimization methods.

Characterizing and optimizing penalty parameters in QUBO formulations is an under-explored area in quantum optimization. In earlier work, Boros and Hammer (2002) introduced the sum of coefficients method for pseudo-Boolean optimization problems and refined it through posiform transformations of QUBO formulations (Boros et al. (2006)). In the context of quantum computing, initial characterizations of penalty parameters for various QUBO problems are provided by Lucas (2014) and Glover et al. (2022). For knapsack problems specifically, Quintero and Zuluaga (2021) offered a characterization of penalty parameters, while recent studies by Awasthi et al. (2023) explored quantum computing techniques for the MKP without focusing on the characterization of the penalty parameters, where similar penalty coefficients to our work are proposed.

Further, the work by Verma and Lewis (2022) presents a general method for determining effective penalty parameter values in QUBO problems. Assuming minimization problems, the authors developed a heuristic to derive suitable lower bounds for penalty parameters. Building on this, García et al. (2022) introduced a sequential algorithm to refine penalty parameters and validated their approach on classical optimization problems. These studies provide a foundation for our research into penalty parameter characterization in quantum optimization.

Lastly, the 0/1 MKP, characterized by discrete variables and multiple constraints, naturally aligns with the QUBO formulations. This property, coupled with the non-fractional structure of the problem, makes it an ideal candidate for exploring the potential of quantum optimization. Beyond its intrinsic optimization challenges, the MKP may also serve as a benchmark for platform-agnostic testing of optimization techniques on the competing current and near-term quantum device architectures (Dam et al. (2021)).

This work contributes to the foundational understanding of applying quantum computing to the MKP by:

- Developing a QUBO formulation for the 0/1 MKP, addressing practical scenarios and con-

straints to enhance the applicability and accuracy of these models in quantum settings.

- Providing a detailed analysis of penalty parameters, which are used to encode constraints within MKP-QUBO models. We expect the proper characterization of these parameters to impact the feasibility of solutions and the efficiency of optimization algorithms.
- Conducting simulation experiments to evaluate the proposed formulations on current quantum platforms. These experiments offer insights into the practical performance and limitations of the models across different quantum computing platforms, providing benchmarks against a current state-of-the-art solver and for future studies.

The remainder of the paper is structured as follows: Section 2 outlines the mathematical formulation of the 0/1 MKP and the characterization of penalty parameters essential for constraint encoding. Section 3 presents the quantum optimization methods employed in this study. Section 4 describes the experimental setup and discusses the results obtained from implementing the formulations on available quantum devices. Finally, Section 5 concludes with a summary of findings and potential directions for future research.

2 MATHEMATICAL MODELS

The 0/1 MKP is among the most extensively studied variations of the classical knapsack problem, distinguished by involving multiple knapsacks rather than a single one. The objective is to allocate a set of items across multiple knapsacks, each with its own capacity, to maximize the total profit while ensuring the weight assigned to any knapsack does not exceed its capacity. Formally, given a set of items \mathcal{N} , where each item has a profit c_{ik} specific to a particular knapsack and a weight d_i , and a set of knapsacks \mathcal{K} , each with a capacity E_k , the problem is to determine the optimal distribution of items across the knapsacks. To achieve this, binary decision variables x_{ik} are introduced, where $x_{ik} = 1$ if item i is assigned to knapsack k , and $x_{ik} = 0$ otherwise. Without loss of generality, all parameters c , d , and E are assumed to be non-negative integers.

The 0/1 MKP can be mathematically formulated as follows:

MKP:

$$\max f(x) = \sum_{k=1}^K \sum_{i=1}^N c_{ik} x_{ik} \quad (1)$$

$$\text{s.t. } \sum_{i=1}^N d_i x_{ik} \leq E_k, \quad k \in \mathcal{K} \quad (2)$$

$$\sum_{k=1}^K x_{ik} \leq 1, \quad i \in \mathcal{N} \quad (3)$$

$$x_{ik} \in \{0, 1\}, \quad i \in \mathcal{N}, k \in \mathcal{K} \quad (4)$$

In this formulation, the objective function (1) is designed to maximize the total profit of the selected items. The knapsack capacity constraints (2) ensure that the total weight assigned to each knapsack k does not exceed its capacity E_k . Additionally, a second set of constraints (3) prevents any item from being placed in more than one knapsack. Finally, the binary restrictions on the decision variables are specified in (4).

To construct the QUBO formulation of the MKP, slack variables $u \in \{0, 1\}^{E_k \times K}$ are introduced for the constraints in (2), while another set of slack variables $v \in \{0, 1\}^N$ is introduced for the constraints in (3). These constraints are then incorporated into the objective function by scaling them with penalty parameters λ_1 and λ_2 , respectively, resulting in the penalty function $P(x, u, v)$.

$$P(x, u, v) = \lambda_1 \left[\sum_{k=1}^K \left(\sum_{i=1}^N d_i x_{ik} - \sum_{t=0}^{M_k-1} 2^t u_{tk} - \alpha_k u_{M_k k} \right)^2 \right] + \lambda_2 \left[\sum_{i=1}^N \left(\sum_{k=1}^K x_{ik} + v_i - 1 \right)^2 \right] \quad (5)$$

Here, $M_k = \lfloor \log_2 E_k \rfloor$ and $\alpha_k = E_k + 1 - 2^{M_k}$, $k \in \mathcal{K}$. It is important to note that when the polynomials are expanded, the penalty function $P(x, u, v)$ includes a constant term $N\lambda_2$ due to the 1 present in the second penalty term. However, as the removal of constant terms does not affect the optimal solution, we proceed under the assumption that $P(x, u, v)$ contains only variable-dependent terms, with the constant term omitted. The resulting QUBO formulation for the MKP is presented below.

MKP-QUBO:

$$\max g(x, u, v) = \sum_{k=1}^K \sum_{i=1}^N c_{ik} x_{ik} - P(x, u, v) \quad (6)$$

$$x_{ik}, u_{tk}, v_i \in \{0, 1\}, i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{M}_k \quad (7)$$

Theorem 1. For any penalty constant λ_1 and λ_2 such that $\lambda_1 \geq C^*$ and $\lambda_2 \geq C^*$, where $C^* = \max\{c_{ik} : i \in \mathcal{N}, k \in \mathcal{K}\}$, (MKP-QUBO) is a valid reformulation of the 0/1 multi-knapsack problem (MKP).

Proof. Let x^* be an optimal solution for (MKP). Let α denote the set of coefficients for u in its binary expansion form. Since x^* is a feasible solution for (MKP), there exists u^* such that $d^T x^* = \alpha_t^T u^*$ for each $k \in \mathcal{K}$, satisfying the constraints in (2). Similarly, there exists v^* such that the constraints in (3) are satisfied, leading to $g(x^*, u^*, v^*) = c^T x^* = f(x^*)$.

Now consider $(\hat{x}, \hat{u}, \hat{v})$, an optimal solution to (MKP-QUBO). By construction, $g(\hat{x}, \hat{u}, \hat{v}) \geq g(x^*, u^*, v^*) = f(x^*)$, as (x^*, u^*, v^*) is also a feasible solution to (MKP-QUBO).

To complete the proof, we need to demonstrate that $g(\hat{x}, \hat{u}, \hat{v}) \leq g(x^*, u^*, v^*) = f(x^*)$ by establishing that \hat{x} is also a feasible solution to (MKP). We will do so by contradiction, showing that an infeasible solution x' , which is feasible for MKP-QUBO but not for MKP, cannot yield a higher objective value. In this sense, our proof strategy differs from the one in (Quintero and Zuluaga (2021)). Assume x^* is the optimal solution to (MKP). We construct an infeasible solution x' for (MKP) and demonstrate that $g(x', u', v') \leq g(x^*, u^*, v^*) = f(x^*)$ always holds.

Select an index pair (j, m) from the complement of the support of x^* , that is, $(j, m) \in \text{supp}^c(x) = \{(i, k) \mid x_{ik}^* = 0\}$. Set $x_{jm} = 1$. By doing so, we are adding an extra item j to knapsack m , which results in an infeasible solution for (MKP). Let x' denote this modified solution.

Next, we construct the corresponding (MKP-QUBO) objective by isolating x_{jm} in the objective function. We explicitly display the penalty terms associated with the m -th and j -th constraints, while introducing P^C to represent the remaining penalty terms of P :

Set $x_{jm} = 1$. In this case, we are adding an extra item j into knapsack m , which should result in an infeasible solution for (MKP). Let's represent the new (extended) solution as x' . Next, one can construct the following (MKP-QUBO) by simply separating x_{jm} from the objective function and also displaying explicitly the m -th and j -th penalty terms corresponding to respective constraints and thus introducing P^C as the rest of the penalty function P :

$$\begin{aligned} g(x', u', v') &= \sum_{k=1}^K \sum_{i=1}^N c_{ik} x'_{ik} - \lambda_1 \sum_{k=1}^K \left(\sum_{i=1}^N d_i x'_{ik} - \sum_{t=0}^{M_k} \alpha_t u'_{tk} \right)^2 \\ &\quad - \lambda_2 \sum_{i=1}^N \left(\sum_{k=1}^K x'_{ik} + v'_i - 1 \right)^2 \\ &= g(x^*, u^*, v^*) + c_{jm} x'_{jm} - \lambda_1 \left(\sum_{i=1}^N d_i x'_{im} - \sum_{t=0}^{M_m} \alpha_t u'_{tm} \right)^2 \\ &\quad - \lambda_2 \left(\sum_{k=1}^K x'_{jk} + v'_j - 1 \right)^2 - P^C(x', u', v') \end{aligned}$$

Since x^* is optimal for (MKP), it follows that $f(x^*) = g(x^*, u^*, v^*)$. Additionally, $P^C(x', u', v') = 0$, as it does not include any terms involving x_{jm} and corresponds to constraints that are already satisfied.

Let us define F as the sum of the additional terms in the last equation, excluding $g(x^*, u^*, v^*)$ and $P^C(x', u', v')$. The key question is whether, with the current choices for λ_1 and λ_2 , it always holds that $F \leq 0$. This ensures that $g(x', u', v') \leq g(x^*, u^*, v^*)$, thereby preserving the optimality of the solution (x^*, u^*, v^*) for (MKP-QUBO).

Since x' is infeasible for (MKP), it must violate either the m -th knapsack capacity constraint or the j -th knapsack choice constraint, or both. Let us first assume that only the m -th capacity constraint is violated.

In this case, $\sum_{i=1}^N d_i x'_{im} - \sum_{t=0}^{M_m} \alpha_t u'_{tm} > 0$, with the minimum magnitude of violation occurring when $\sum_{t=0}^{M_m} \alpha_t u'_{tm} = E_m$, which implies $u'_{tm} = 1$ for $t = 0, \dots, M_m$. Under the assumption that the second constraint is not violated, even when $x'_{jm} = 1$, it must be the case that $\sum_{k=1}^K x'_{jk} = 0$. To offset the associated penalty, we set $v'_j = 1$.

When $x'_{jm} = 1$, then $\sum_{k=1}^K x'_{jk} = 1$, and the penalty can once again be offset by setting $v'_j = 0$. Consequently, the term associated with λ_2 can be omitted, and the total change in the objective function is given by:

$$F = c_{jm} - \lambda_1 \left(\sum_{i=1}^N d_i x'_{im} - E_m \right).$$

Given that $\lambda_1 \geq C^*$, the following relationship holds:

$$\lambda_1 \left(\sum_{i=1}^N d_i x'_{im} - E_m \right) \geq \lambda_1 \geq C^* \geq c_{jm} \quad (8)$$

Let us now consider the case where adding the extra item does not cause an overflow in the m -th knapsack capacity constraint but instead results in a violation of the j -th single-choice constraint. In this scenario, we have $(\sum_{i=1}^N d_i x'_{im} - E_m) \leq 0$. Consequently, there exists a solution u' such that $\sum_{t=0}^{M_m} \alpha_t u'_{tm} = \sum_{i=1}^N d_i x'_{im}$, which compensates for the penalty term associated with the first constraint.

The violation of the second constraint, in this case, is exactly 1, as this constraint has not been previously violated due to $x'_{jm} = 0$. Thus, we can express the change in the objective function as $F = c_{jm} - \lambda_2$. We conclude that $F \leq 0$ through the following relationship:

$$\lambda_2 \geq C^* \geq c_{jm}, \quad (9)$$

where C^* represents the maximum profit coefficient.

If both constraints are violated, we can combine the individual results to construct the following relationship, which ensures that $F < 0$:

$$\lambda_1 \left(\sum_{i=1}^N d_i x'_{im} - E_m \right) + \lambda_2 \geq \lambda_1 + \lambda_2 \geq 2C^* > c_{jm}. \quad (10)$$

Thus, when $F < 0$, it follows that $g(x', u', v') > g(x^*, u^*, v^*)$, indicating that an infeasible solution for (MKP) cannot yield a higher objective value in (MKP-QUBO). Consequently, this scenario cannot occur. Therefore, if $\lambda_1 > C^*$ and $\lambda_2 > C^*$, then x^* is both optimal and feasible for (MKP-QUBO).

In the boundary case where $\lambda_1 = C^*$ and $\lambda_2 = C^*$, the same conclusion holds. It is possible to construct a feasible solution $(\hat{x}, \hat{u}, \hat{v})$ from (x', u', v') , indicating that $(\hat{x}, \hat{u}, \hat{v})$ is an alternative optimal solution for (MKP-QUBO). Hence, we have:

$$g(x', u', v') = g(\hat{x}, \hat{u}, \hat{v}) = c^T \hat{x} \leq c^T x^* = f(x^*). \quad \square$$

An interesting observation arises in the third scenario, where both constraints are violated. In this case, the condition requiring each penalty parameter to be at least C^* may be overly restrictive. The coefficient of λ_1 can be as large as $d^* = \max\{d_i : i \in N\}$. Therefore, as long as $d^* \lambda_1 + \lambda_2 \geq C^*$, both formulations will produce the same optimal solution, which depends heavily on the problem data.

3 SOLUTION METHODOLOGY

To address the MKP instances generated for this study, we employ four distinct solution methods: (i) solving the classical integer linear programming (ILP) formulation of MKP using a commercial solver, (ii) solving the QUBO formulation of MKP with a commercial solver, (iii) utilizing Quantum Annealing, and (iv) applying the Quantum Approximate Optimization Algorithm (QAOA). State-of-the-art commercial solvers are highly effective at solving both the ILP and QUBO formulations of MKP by leveraging advanced operations research techniques. These solvers are mature, well-developed, and fall outside the primary focus of this work.

In the following, we provide a brief overview of these two quantum techniques and their relevance to solving MKP instances.

3.1 Quantum Annealing

Quantum annealing exploits the physical properties of quantum systems to solve optimization problems (Albasha and Lidar (2018)). The underlying principle is rooted in physics: a system will naturally evolve toward its lowest energy state, also known as its ground state (Das and Chakrabarti (2008)). In quantum annealing, a problem is encoded into an entangled quantum system where each qubit represents a binary variable in the QUBO formulation. The system is configured such that its minimum energy state corresponds to the optimal solution of the problem. Upon measurement, the quantum system collapses, with each qubit assuming a binary value of 0 or 1, providing the solution (D-Wave (2022)).

In practice, quantum annealing maps each binary variable of the optimization problem onto a qubit. Interactions between these variables are represented as qubit entanglements, forming a network that encodes the problem's constraints and objective. The goal is to structure the quantum system so that its ground state aligns with the optimal solution of the QUBO. Penalty parameters are used to enforce constraints by assigning higher energy to infeasible solutions, modifying the energy landscape to guide the system toward feasible configurations. These parameters must be set to balance enforcing constraints while preserving the objective function. This ensures that feasible solutions are energetically favorable compared to infeasible ones. However, due to the architectural limitations of quantum hardware, such as restricted qubit connectivity, additional qubits may be required to embed the problem, which increases resource demands.

The annealing process begins by initializing the qubits in a superposition state, representing all possible solutions simultaneously. As the system evolves, it is guided toward configurations of progressively lower energy. At the conclusion of the process, the system is measured, causing the qubits to collapse into a classical state of either 0 or 1. Each combination of these states has an associated energy, which reflects the value of the QUBO objective function. The solution with the lowest energy corresponds to the optimal configuration.

The energy landscape of a quantum annealer is a visualization of the energy values across all possible configurations. For instance, Figure 1 illustrates the energy landscape for a hypothetical QUBO instance with four binary variables, resulting in $2^4 = 16$ possible solutions. The height of each state in the landscape represents its corresponding energy, including contributions from penalty terms that distinguish feasible and infeasible solutions. States such as 0011 and

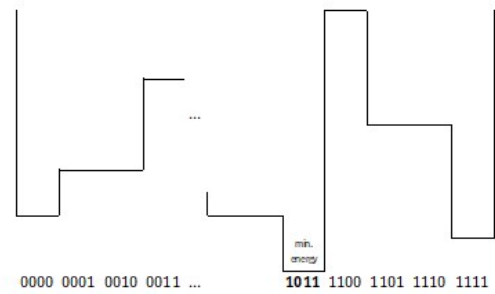


Figure 1: Energy landscape illustration for a hypothetical QUBO instance with 4 binary variables.

1100 are shown with higher energy, indicating suboptimal solutions.

3.2 Quantum Approximate Optimization Algorithm (QAOA)

The Quantum Approximate Optimization Algorithm (QAOA) is a widely studied variational quantum algorithm developed to tackle combinatorial optimization problems that are challenging for classical methods (Blekos et al. (2024)). QAOA falls under the category of hybrid quantum-classical algorithms and operates on gate-based quantum computers. The algorithm constructs a parameterized quantum circuit using a sequence of quantum gates that encode the optimization problem. This circuit alternates between two types of gates, each associated with a specific Hamiltonian: (i) the Cost Hamiltonian, which represents the objective function (e.g., a QUBO problem), and (ii) the Mixer Hamiltonian, which facilitates exploration of the solution space. Together, these gates form a single "layer" of the QAOA circuit, and multiple layers can be stacked to improve performance (Farhi et al. (2014)).

QAOA is designed to run on gate-based quantum devices, including those based on superconducting qubits (such as IBM and Rigetti systems) or trapped ions (such as IonQ). The algorithm introduces two tunable parameters, γ (associated with the Cost Hamiltonian) and β (associated with the Mixer Hamiltonian), which are iteratively refined using a classical non-linear optimization routine. γ determines the system's alignment with the objective function, influencing the depth of energy minimization, while β controls the degree of exploration in the solution space. Together, these parameters guide the system toward an optimal configuration by balancing exploitation and exploration. This iterative process uses feedback from quantum measurements to adjust the circuit parameters, forming a hybrid algorithm that integrates classical optimization with quantum ex-

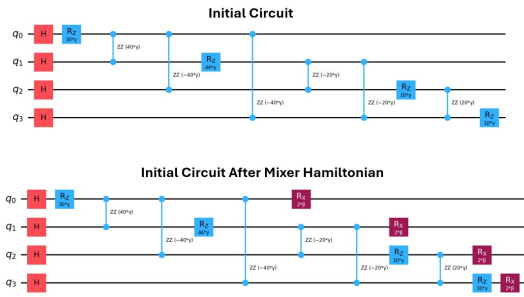


Figure 2: Quantum circuit used in QAOA.

cution. While the quantum component (circuit execution) is fully gate-based, the overall process incorporates a classical optimization loop (Farhi et al. (2014)).

Figure 2 illustrates the quantum circuit for a small QUBO instance with four binary variables. The problem data is directly encoded into the circuit through specific rotations determined by the coefficients of the QUBO terms. The Mixer Hamiltonian is then applied, completing one iteration of the QAOA process. Following the optimization of the parameters γ and β , the circuit is sampled, ideally with a large number of shots, to identify the configuration that minimizes the system’s energy with the expectation of this configuration to yield the optimal solution of the original optimization problem. These parameters hence affect the quantum state’s evolution during the circuit execution, shaping the energy landscape such that the system is steered toward low-energy configurations corresponding to optimal or near-optimal solutions.

QAOA includes several adjustable settings, such as the number of circuit layers, the method for initializing parameters, and the choice of classical optimization technique. The actual circuits used in quantum hardware are often more complex than the illustrative example provided here. These variations in settings and implementation details significantly impact the performance of the algorithm on practical optimization problems.

4 EXPERIMENTAL ANALYSIS

This section examines how penalty parameter calibration influences the performance of quantum optimization methods for the MKP. Experiments are conducted using a simulation-based approach, enabling controlled testing of parameter adjustments under practical constraints. By generating diverse problem instances and varying penalty parameter values, we analyze their impact on solution feasibility, quality, and computational efficiency across different quan-

tum technologies, with classical optimization results provided for comparison.

4.1 Testbed

Our testbed comprises randomly generated MKP instances, with the number of items set to $N \in \{3, 4, 5, 6\}$ and the number of knapsacks varying as $K \in \{2, 3\}$. The revenue values c are randomly chosen as integers in the range $[1, 10]$, while the item weights d are selected as random integers within $[1, 5]$. The capacities of the knapsacks, E , are also generated randomly as integers between 60% and 80% of the total weight of items assignable to that knapsack. This range ensures that not all items can be placed simultaneously.

For the penalty parameters λ , the default value is set as $\lambda = C^* = \max\{c_{ik}\}$. To evaluate the impact of penalty parameter scaling, a multiplier ranging from 0.5 to 1.5 is applied to λ , with increments of 0.02, generating a total of 50 distinct λ values. This setup allows us to investigate the potential for smaller λ values to cause infeasibility, as well as to analyze how the magnitude of λ affects solution times.

In total, $4 \times 2 \times 50 = 300$ problem instances are created. Note that these instances are relatively small compared to the size of MKP instances typically solvable by classical computers. However, due to the limited number of qubits available in current Noisy Intermediate-Scale Quantum (NISQ) quantum devices, solving larger problem instances efficiently is not feasible with the present state of quantum technology.

4.2 Experimental Setup and Implementation Details

To generate the problem instances, we developed a C# console application using Microsoft Visual Studio Community 2022. Both the classical MIP formulation and the QUBO formulation of the MKP instances were created using Gurobi callable libraries and solved using the Gurobi 12.0 Solver (Gurobi Optimization, LLC (2024)). For the quantum implementations, the instances were tested across three primary quantum technologies: gate-based circuits, trapped-ion circuits, and quantum annealing.

For the quantum annealing experiments, we utilized D-Wave libraries (D-Wave (2022)). Using D-Wave’s Python interface, the QUBO formulation of each instance was transformed into an optimization problem and solved using the annealer. For gate-based quantum experiments, we relied on IBM Qiskit libraries, where the Q matrix representing the co-

efficients of the QUBO instance was used to construct quantum circuits. A standard QAOA subroutine was then executed to find the best solution. In the QAOA implementation, the circuit depth is chosen to be $p = 3$ which is the recommended value in many works (Blekos et al. (2024)). Qiskit implementation of QAOA uses the COBYLA solver (Powell (1994)) as the default, non-linear, derivative-free solver to optimize the circuit parameters. Finally, the trapped-ion experiments employed IonQ Python libraries. Similar to the gate-based approach, circuits were generated from the Q matrix of each instance and solved using the QAOA algorithm.

Each instance is run 10 times on the quantum platforms, with a sample size (or number of shots) set to 10,000. For both annealing and gate-based approaches, increasing the number of shots mitigates errors inherent to quantum mechanics, improving solution accuracy. In every experiment, we recorded the best solution, the corresponding objective value, and the solution time. Since the MIP solver reliably finds the optimal solutions for both the classical MIP and QUBO formulations, these serve as a benchmark. Additionally, we assessed whether the QUBO solutions obtained from the solver and the three quantum-based methods were feasible and/or optimal.

4.3 Computational Results

We began by comparing the running times of the five methods tested in our experiments, as summarized in Table 1. These times represent averages over $10 \times 50 = 500$ runs (10 randomly generated instances with 50 different values of the penalty parameter) for each combination of N and K .

In Table 1, the first column lists the number of items (N) and knapsacks (K) for the generated problem instances. The next two columns show the average solution times obtained using Gurobi for the classical integer programming (MIP) and QUBO formulations of the MKP. The final three columns present the average solution times for the quantum-based methods.

As expected, Gurobi achieved the fastest solution times for the classical integer programming formulation of the MKP, followed closely by the QUBO version solved with Gurobi. The D-Wave quantum annealer demonstrated comparable performance to Gurobi’s quadratic solver, solving nearly all instances in under one second. It is notable that D-Wave’s average running time is smaller than Gurobi’s quadratic solver for the largest instances corresponding to 5/6 item and 3 knapsack scenarios. In contrast, QAOA-based methods were significantly slower, with solu-

tion times increasing as the problem size grew. For the largest instances both annealing methods fail to provide solutions either with out of memory or time-out errors.

For the IonQ simulator, instances with $(N, K) = (4, 3)$, $(5, 3)$, and $(6, 3)$ could not be solved due to memory limitations, highlighting current hardware constraints for larger problem sizes.

Table 1: Comparison of running times across different solvers for varying values of N (number of items) and K (number of knapsacks).

N,K	Int. Pr. Gur.		Qu. Annl.	QAOA	
	IP	QUBO	D-Wave	Qiskit	IonQ
3,2	0.003	0.06	0.33	23.1	17.9
4,2	0.003	0.14	0.36	30.5	33.8
5,2	0.012	0.22	0.51	184.1	276.8
6,2	0.124	0.42	0.67	641.1	2075.4
3,3	0.005	0.07	0.40	86.9	209.3
4,3	0.007	0.10	0.44	371.6	-
5,3	0.005	1.15	0.54	-	-
6,3	0.005	2.56	0.56	-	-

Table 2 compares the solution quality of the quantum-based methods. Since the optimal solutions for all instances are known, we evaluate the performance of the three quantum-based methods by calculating the percentage of instances where they successfully find the optimal solution.

D-Wave achieves the optimal solution in almost all cases for the smaller size instances but may decrease up to 54.6% for most complex cases. For smaller instances, QAOA-based methods also perform well, with IonQ generally outperforming Qiskit. However, as the instance size increases, the solution quality of the QAOA-based methods declines significantly.

Table 2: Per-cent of Optimal Solutions found by the quantum-based method.

N,K	Qu. Annl.	QAOA	
	D-Wave	Qiskit	IonQ
3,2	100.0%	85.3%	92.7%
4,2	99.7%	86.0%	95.9%
5,2	99.8%	71.9%	85.8%
6,2	99.4%	54.8%	66.8%
3,3	99.2%	63.8%	69.0%
4,3	99.5%	42.2%	-
5,3	68.6%	-	-
6,3	54.6%	-	-

4.4 Effect of Penalty Parameter Magnitude on Feasibility

In the proof of Theorem-1, we noted that requiring the penalty parameter λ to be at least $\lambda = C^* = \max\{c_{ik}\}$

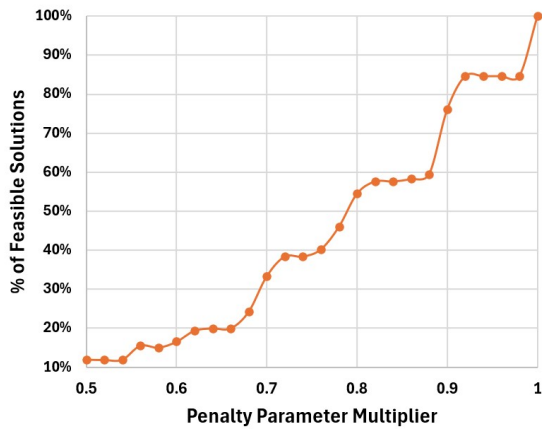


Figure 3: Effect of penalty parameter magnitude on solution feasibility.

may be overly restrictive for certain instances. To investigate this, we calibrated the value of λ using a multiplier ranging from 0.5 to 1.5. Figure 3 illustrates the average percentage of instances that resulted in feasible solutions under these varying penalty parameter settings.

Each data point in the figure represents the average results over $10 \times 8 = 80$ instances (10 random instances across 8 different combinations of N and K). When the penalty parameter coefficient is set to 1.00, λ equals C^* , and all QUBO instances produce the same optimal solutions as the original integer programming formulation of the MKP.

However, as the penalty parameter is reduced below this theoretical minimum, a gradual decline in feasibility is observed, with fewer QUBO solutions satisfying the constraints of the original MKP. Future research could focus on deriving tighter bounds for penalty parameters tailored to specific MKP instances, potentially reducing the need for conservative parameter settings.

4.5 Effect of Penalty Parameter Magnitude on Solution Duration

Finally, we evaluate how the magnitude of the penalty parameter affects the running time of the solution methods. This analysis is based on normalized running times for 80 instances across 4 QUBO-based methods. The results, summarized in the box-whisker plot in Figure 4, indicate that the magnitude of the penalty parameter has minimal impact on solution times. A consistent pattern is observed across all individual quantum methods (e.g., annealing and QAOA); hence, more granular plots for specific methods are omitted.

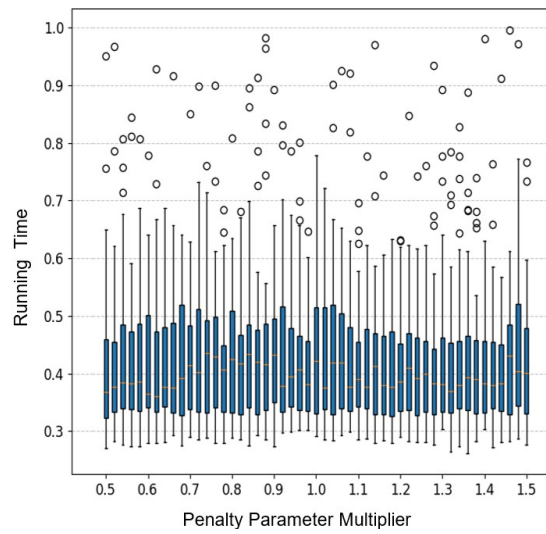


Figure 4: Effect of penalty parameter magnitude on running time.

Note that these results were obtained using classical quantum system simulators, which may not fully capture the computational dynamics of actual quantum hardware. Replicating these experiments on real quantum devices would provide more accurate insights and validate the observations made here.

5 CONCLUSIONS

This study investigates the application of QUBO formulations to the 0/1 Multi-Knapsack Problem (MKP) using classical and quantum techniques.

Specifically, we characterize and analyze the penalty parameters, demonstrating that overly conservative settings can impact feasibility without significantly affecting solution times. Our results highlight the strengths and limitations of quantum annealing and QAOA in solving MKP instances on simulators, with annealing showing potential for handling larger problem sizes, albeit with suboptimal solutions. Specifically for our investigation, a state-of-the-art classical solver remains superior.

Further, our research suggests exploring better ways to configure penalty parameters for annealing, as well as simplifying quadratic representations to reduce computational overhead across all platforms, both quantum and classical. Lastly, testing across a wider range of simulators and on real quantum devices may further guide scholarly efforts in adopting quantum techniques for common operations research problems.

REFERENCES

- Albash, T. and Lidar, D. A. (2018). Demonstration of a scaling advantage for a quantum annealer over simulated annealing. *Physical Review X*, 8(3):031016.
- Arute, F. et al. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510.
- Awasthi, A., Bär, F., Doetsch, J., Ehm, H., Erdmann, M., Hess, M., Klepsch, J., Limacher, P. A., Luckow, A., Niedermeier, C., Palackal, L., Pfeiffer, R., Ross, P., Safi, H., Schönmeier-Kromer, J., von Sicard, O., Wenger, Y., Wintersperger, K., and Yarkoni, S. (2023). Quantum computing techniques for multi-knapsack problems. In Arai, K., editor, *Intelligent Computing*, pages 264–284. Springer Nature Switzerland.
- Blekos, K., Brand, D., Ceschini, A., Chou, C.-H., Li, R.-H., Pandya, K., and Summer, A. (2024). A review on quantum approximate optimization algorithm and its variants. *Physics Reports*, 1068:1–66. A review on Quantum Approximate Optimization Algorithm and its variants.
- Boros, E. and Hammer, P. L. (2002). Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1–3):155–225.
- Boros, E., Hammer, P. L., and Tavares, G. (2006). Preprocessing of unconstrained quadratic binary optimization. Technical Report RRR 10-2006, Rutgers University, New Jersey, USA. RUTCOR Research Report.
- D-Wave (2022). What is quantum annealing? Accessed: 2024-11-11.
- Dam, W. v., Eldefrawy, K., Genise, N., and Parham, N. (2021). Quantum optimization heuristics with an application to knapsack problems. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 160–170. IEEE.
- Das, A. and Chakrabarti, B. K. (2008). Colloquium: Quantum annealing and analog quantum computation. *Rev. Mod. Phys.*, 80:1061–1081.
- Farhi, E., Goldstone, J., and Gutmann, S. (2014). A quantum approximate optimization algorithm.
- García, M. D., Ayodele, M., and Moraglio, A. (2022). Exact and sequential penalty weights in quadratic unconstrained binary optimisation with a digital annealer. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '22*, page 184–187, New York, NY, USA. Association for Computing Machinery.
- Glover, F., Kochenberger, G., and Du, Y. (2019). A tutorial on formulating and using qubo models. *arXiv preprint*, arXiv:1811.11538.
- Glover, F., Kochenberger, G., Hennig, R., Lewis, M., Lü, Z., Wang, H., and Wang, Y. (2022). Quantum bridge analytics i: a tutorial on formulating and using qubo models. *Annals of Operations Research*, 314:141–183.
- Gurobi Optimization, LLC (2024). Gurobi Optimizer Reference Manual.
- Lucas, A. (2014). Ising formulations of many np problems. *Frontiers in Physics*, 2.
- Pecyna, T. and Różycki, R. (2024). Improving quantum optimization algorithms by constraint relaxation. *Applied Sciences*, 14(18):8099.
- Powell, M. J. D. (1994). A direct search optimization method that models the objective and constraint functions by linear interpolation. *Advances in Optimization and Numerical Analysis*, pages 51–67.
- Pusey-Nazzaro, L. and Date, P. (2020). Adiabatic quantum optimization fails to solve the knapsack problem. Accessed: 2024-12-04.
- Quintero, R. A. and Zuluaga, L. F. (2021). Characterizing and benchmarking qubo reformulations of the knapsack problem. Technical report, Department of Industrial and Systems Engineering, Lehigh University. Technical Report.
- Verma, A. and Lewis, M. (2022). Penalty and partitioning techniques to improve performance of qubo solvers. *Discrete Optimization*, 44:100594. Quadratic Combinatorial Optimization Problems.