IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 February 2025, accepted 28 February 2025, date of publication 12 March 2025, date of current version 21 March 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3550788

RESEARCH ARTICLE

QUBO Formulations and Characterization of Penalty Parameters for the Multi-Knapsack Problem

EVREN GÜNEY[1], **JOACHIM EHRENTHAL**[2], **AND THOMAS HANNE**[2]
[1]Department of Industrial Engineering, MEF University, 34396 İstanbul, Türkiye
[2]Institute of Business Information Systems, School of Business, University of Applied Sciences and Arts Northwestern Switzerland FHNW, 5210 Windisch, Switzerland

Corresponding author: Evren Güney (guneye@mef.edu.tr)

ABSTRACT The Multi-Knapsack Problem (MKP) is a fundamental challenge in operations research and combinatorial optimization. Quantum computing introduces new possibilities for solving MKP using Quadratic Unconstrained Binary Optimization (QUBO) models. However, a key challenge in QUBO formulations is the selection of penalty parameters, which directly influence solution feasibility and algorithm performance. In this work, we develop QUBO formulations for two MKP variants—the Multidimensional Knapsack Problem (MDKP) and the Multiple Knapsack Problem (MUKP)—and provide an algebraic characterization of their penalty parameters. We systematically evaluate their impact through quantum simulation experiments and compare the performance of the two leading quantum optimization approaches: Quantum Approximate Optimization Algorithm (QAOA) and quantum annealing, alongside a state-of-the-art classical solver. Our results indicate that while classical solvers remain superior, careful tuning of penalty parameters has a strong impact on quantum optimization outcomes. QAOA is highly sensitive to parameter choices, whereas quantum annealing produces more stable results on small to mid-sized instances. Further, our results reveal that MDKP instances can maintain feasibility at penalty values below theoretical bounds, while MUKP instances show greater sensitivity to penalty reductions. Finally, we outline directions for future research in solving MKP, including adaptive penalty parameter tuning, hybrid quantum-classical approaches, and practical optimization strategies for QAOA, as well as real-hardware evaluations.

INDEX TERMS Multi-dimensional knapsack problem, multiple knapsack problem, quadratic unconstrained binary optimization, quantum annealing, quantum approximate optimization algorithm, penalty parameters.

I. INTRODUCTION

The Multi-Knapsack Problem (MKP) is a fundamental combinatorial optimization problem that generalizes the classic knapsack problem to multiple knapsacks. The MKP is classified as an NP-hard problem and holds importance in operations research due to its wide range of practical applications across various industries where efficient resource allocation is critical, such as logistics, production, and finance [1]. Classical, i.e. non-quantum, optimization approaches, including exact algorithms, heuristics, and metaheuristics, have been highly effective in solving many MKP instances, enabling robust solutions for both small-scale and large-scale problems. However, as the landscape of optimization problems evolves and problem sizes increase, exploring alternative approaches such as quantum computing remains valuable to assess their potential contributions alongside classical methods [2].

Recent advances in quantum computing offer new approaches to combinatorial optimization problems like the MKP. Quantum computing exploits quantum mechanical principles, such as superposition and entanglement, allowing the exploration of solution spaces through different mechanisms compared to classical approaches. Many quantum

The associate editor coordinating the review of this manuscript and approving it for publication was Guillermo Botella Juan.

devices primarily support Quadratic Unconstrained Binary Optimization (QUBO) models, making them suitable for exploratory research on problems like the MKP [3], [4].

A key difference between classical techniques and QUBO lies in how constraints are handled. Classical techniques explicitly model constraints to ensure feasibility and computational efficiency. In contrast, QUBO integrates constraints directly into the objective function using penalty parameters, requiring careful calibration. Penalty parameters play a key role in enforcing constraints in QUBO formulations. They must be large enough to prevent constraint violations but not so large that they distort the optimization landscape. If penalties are too small, solutions may violate feasibility conditions. Conversely, excessive penalty values introduce large energy gaps that hinder search behavior [5], [6]. However, the penalty behavior is platform-dependent, where the two dominant quantum computing platforms for optimization problems are annealing and gate-based methods.

Quantum annealers solve QUBO problems by gradually minimizing an energy function, making them naturally suited for these formulations but susceptible to analog control imprecisions. Gate-based methods, which predominantly use QAOA for combinatorial optimization, encode QUBO problems using quantum circuits, where penalty terms directly affect circuit depth and noise resilience. As QAOA is executed through discrete gate operations, higher penalty values translate into increased gate count, worsening decoherence effects and reducing solution accuracy [7], [8].

More specifically, quantum annealers, such as those developed by D-Wave, evolve a quantum system toward its lowest-energy configuration [9]. Logical variables must be embedded into the hardware topology, such as Chimera or Pegasus graphs, increasing qubit requirements and introducing additional noise [10]. Unlike gate-based quantum computers, annealers use continuous control signals to encode problem coefficients, which introduces rounding errors in penalty terms and affects feasibility [11]. Additionally, decoherence and thermal noise degrade solution quality by disturbing the quantum state during optimization [12], [13]. The annealing schedule also influences performance: if the system evolves too quickly, it may settle into local minima rather than the optimal solution [14]. However, since D-Wave operates natively on QUBO formulations, it provides a direct testbed for MKP instances, yet penalty parameters must be carefully set to ensure constraint enforcement while avoiding energy barriers that hinder exploration of the solution space.

Gate-based quantum computing platforms, such as IBM Qiskit (superconducting qubits) and IonQ (trapped-ion qubits), implement quantum optimization methods like the Quantum Approximate Optimization Algorithm (QAOA) to solve QUBO problems [15], [16]. Unlike quantum annealers, these systems rely on discrete gate operations, requiring a different approach to constraint handling. Moreover, IBM Qiskit simulates superconducting qubits, where limited qubit connectivity forces additional SWAP operations to enable interactions between non-adjacent qubits [17]. These additional operations increase circuit depth, amplifying decoherence effects and reducing solution accuracy [18]. In contrast, trapped-ion quantum computers, such as those developed by IonQ, offer all-to-all connectivity, eliminating the need for SWAP gates [16]. However, trapped-ion platforms have significantly slower gate execution times compared to superconducting architectures, which limits scalability [19]. Also, QAOA introduces further complexity in constraint enforcement, as penalty parameters impact not only the optimization landscape but also the feasibility of circuit execution. QAOA depends on tunable parameters γ and β, which dictate how the quantum state evolves. These parameters must be carefully optimized, as no analytical selection rules exist, making QAOA effectivenes performance dependent on initialization strategies and classical optimization heuristics [20]. Increasing penalty values requires additional gate operations, deepening circuits and amplifying decoherence effects. This makes penalty selection in QAOA highly sensitive to both problem characteristics and the hardware used [5], [6].

The characterization and optimization of penalty parameters remain underexplored in quantum optimization. Boros and Hammer [21] introduced the sum of the coefficients method for pseudo-Boolean optimization problems and later refined it using posiform transformations [22]. Basic penalty parameter characterizations for QUBO formulations were provided by Lucas [3] and Glover et al. [4]. For knapsack problems specifically, Quintero and Zuluaga [23] offer a detailed analysis of penalty parameters, while Awasthi et al. [24] explore quantum computing techniques for the multi-knapsack problem without addressing penalty selection. A more general approach was proposed by Verma and Lewis [5], who developed a heuristic method to derive lower bounds for penalty parameters in QUBO models. Building on this, García et al. [6] introduced a sequential search method to optimize penalty values, testing their approach across various classical optimization problems.

Building on these foundations, our work advances the application of quantum computing to the MKP by:

- Developing QUBO formulations for two MKP variants: the 0/1 Multi-Dimensional Knapsack Problem (MDKP) and the 0/1 Multiple Knapsack Problem (MUKP).
- Providing a mathematical characterization of penalty parameters for encoding constraints within MKP-QUBO formulations.
- Conducting experiments to evaluate the proposed formulations using quantum simulators for D-Wave, IBM Qiskit, and IonQ, offering an approachable assessment of performance on actual quantum devices.

The remainder of the paper is organized as follows. Section II introduces the mathematical model of the two MKP variants and the characterization of penalty parameters essential for constraint encoding. In Section III, details of the quantum optimization methods are presented. Section IV describes the experimental setup and reports the results of implementing the formulations on quantum simulators.

Finally, Section V concludes with a discussion of the findings and directions for future research.

II. MATHEMATICAL MODELS

In this section, we present classical mathematical formulations of two multi-knapsack problem variants: Multi-dimensional Knapsack Problem (MDKP) and Multiple Knapsack Problem (MUKP) together with their corresponding QUBO formulations. In each of these formulations, some common notation will be used. The set of items is represented with $\mathcal{N}$ of size $|\mathcal{N}| = N$, whereas the set of knapsacks is represented with $\mathcal{K}$ of size $|\mathcal{K}| = K$. For the multi-dimensional knapsack problem we define the set of dimensions $\mathcal{D}$ with size $|\mathcal{D}| = D$. The revenue acquired from placing an item to a knapsack is represented by parameters r_i and the amount of resource consumed by item i is w_i. Lastly, the capacity of knapsack k is shown by parameters W_k. These parameters may be used in multi-dimensional forms if there are more than one dependency (e.g. r_{ik} for revenue earned by item i placed on knapsack k), which will be clarified in the corresponding model. Without loss of generality we assume all the problem parameters r, w, W to be integers.

A. 0/1 MULTI-DIMENSIONAL KNAPSACK PROBLEM (MDKP)

In MDKP there are N items with revenues $r_i > 0$ and a single knapsack with multiple dimensions of quantity D with capacities W_d. Each item i occupies an amount of $w_{id} \geq 0$ in the d-th dimension of the knapsack. We define binary variables x_i to represent if item i is being selected or not. The objective is to maximize the sum of profits of the selected items so that the sum of weights consumed in each dimension does not exceed W_d. Then the 0/1 MDKP could be expressed with the following integer programming formulation:

MDKP:

$$\max \sum_{i=1}^{N} r_i x_i \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{N} w_{id} x_i \leq W_d, \quad d \in \mathcal{D} \tag{2}$$

$$x_i \in \{0, 1\}, \quad i \in \mathcal{N} \tag{3}$$

In this formulation the objective function (1) maximizes the total revenue. Constraint set (2) are the capacity constraints corresponding to each separate dimension of the knapsack (or in a more generic sense, resource d). Last, the formulation ends with binary requirements (3) on the decision variables x_i. If $D = 1$ then, MDKP reduces to the classical 0/1 Knapsack Problem. Without loss of generality, we can assume that $w_{id} \leq W_d$ for all dimensions d for each item i. Otherwise item i can be discarded from the formulation. Also, we can assume that $W_d < \sum_{i=1}^{n} w_{id}$ for at least one dimension d, since otherwise the solution is trivial.

Next, we derive the first QUBO reformulation of the MDKP. We add slack variables s_d to convert inequality constraints to equations to get the following equivalent formulation.

$$\max \sum_{i=1}^{n} r_i x_i \tag{4}$$

$$\text{s.t.} \sum_{i=1}^{N} w_{id} x_i + s_d = W_d, \quad d \in \mathcal{D} \tag{5}$$

$$x_i \in \{0, 1\}, s_d \in \mathbb{Z}^+ \quad i \in \mathcal{N}, d \in \mathcal{D} \tag{6}$$

Then we can rewrite the knapsack constraints as choice constraints by removing the slack variables and introducing binary variables y_{td} as follows:

$$\max \sum_{i=1}^{N} r_i x_i \tag{7}$$

$$\text{s.t.} \sum_{i=1}^{N} w_{id} x_i = \sum_{t=1}^{W_d} t y_{td}, \quad d \in \mathcal{D} \tag{8}$$

$$\sum_{t=1}^{W_d} y_{td} = 1, \quad d \in \mathcal{D} \tag{9}$$

$$x_i, y_{td} \in \{0, 1\}, \quad i \in \mathcal{N}, \quad d \in \mathcal{D}, \quad t \in \mathcal{W}_d \tag{10}$$

Here t is an integer running from 1 to W_d. According to this formulation, for each d, at the optimal solution only the corresponding y_{td} variable will be equal to one and the rest are forced to zero. When all the constraints are carried to the objective function with penalty parameters λ_1 and λ_2 corresponding to the first and second set of constraints, respectively, we get a QUBO reformulation of the original problem as:

$$\max f(x) = \sum_{i=1}^{N} r_i x_i - \lambda_1 \sum_{d=1}^{D} \left(\sum_{i=1}^{n} w_{id} x_i - \sum_{t=1}^{W_d} t y_{td} \right)^2 - \lambda_2 \sum_{d=1}^{D} \left(1 - \sum_{t=1}^{W_d} y_{td} \right)^2 \tag{11}$$

$$\text{s.t. } x_i, y_{td} \in \{0, 1\}, \quad i \in \mathcal{N}, \quad d \in \mathcal{D}, \quad t \in \mathcal{W}_d \tag{12}$$

This QUBO formulation can further be improved and can be represented with less number of slack variables by using a binary representation of the knapsack capacities as shown in [3] and [24]. In the new form, the first penalty term of equation (11) is re-written by binary coefficients on the slack variables and the second penalty term is dropped resulting in the following formulation.

MDKP-QUBO:

$$\max \sum_{i=1}^{n} r_i x_i - \lambda \sum_{d=1}^{D} \left(\sum_{i=1}^{n} w_{id} x_i - \sum_{t=0}^{M_d - 1} 2^t y_{td} - \alpha_d y_{M_d d} \right)^2 \tag{13}$$

$$\text{s.t. } x_i, y_{td} \in \{0, 1\}, \quad i \in \mathcal{N}, \quad d \in \mathcal{D}, \quad t \in \mathcal{M}_d \tag{14}$$

where $M_d = \lfloor log_2 W_d \rfloor$ and $\alpha_d = W_d + 1 - 2^{M_d}$.

Theorem 1: For any penalty constant λ such that $\lambda \geq R^*$, where $R^* = \max\{r_i : i \in \mathcal{N}\}$, the 0/1 multidimensional-knapsack problem (MDKP) can be reformulated as (MDKP-QUBO).

Proof: Let $g(x, y)$ represent the objective function of (MDKP-QUBO) and let x^* be an optimal solution for (MDKP). We introduce the vector α_d to represent the set of coefficients of y in the binary expansion form in constraint d. As x^* is also a feasible solution for (MDKP), then there exists y^*, such that $w_d^T x^* = \alpha_d^T y_d^*$ for every $d \in \mathcal{D}$, satisfying constraint set (2). Hence, there exists a feasible solution (x^*, y^*) for (MDKP-QUBO) such that, $g(x^*, y^*) = f(x^*) = r^T x^*$. Let $(\hat{x}, \hat{y})$ be an optimal solution to (MDKP-QUBO), then $g(\hat{x}, \hat{y}) \geq g(x^*, y^*) = f(x^*)$.

To prove the opposite case, i.e., $g(\hat{x}, \hat{y}) \leq g(x^*, y^*)$, we show by contradiction that $\hat{x}$ is either feasible for (MDKP) or it can never be optimal for (MDKP-QUBO).

Let $g(\hat{x}, \hat{y})$ be an optimal solution for (MDKP-QUBO). Then if $\hat{x}$ is feasible for (MDKP), then there exists y^* such that $w^T \hat{x} = \alpha_t^T y^*$. Consequently,

$$g(\hat{x}, \hat{y}) = g(\hat{x}, y^*) = r^T \hat{x} \leq r^T x^* = f(x^*) \tag{15}$$

The first equality is true, because $\max\{g(\hat{x}, y)\}$ occurs when there is no penalty, meaning $w^T \hat{x} = \alpha_t^T y$. The inequality part is true due to the setting that $\hat{x}$ is already a feasible solution for (MDKP).

Let's create an infeasible solution x' for (MDKP) from the optimal QUBO solution $(\hat{x}, \hat{y})$, by randomly choosing an index (j) from the compliment of the support of $\hat{x}$, formally, $j \in \text{supp}^c(x) = \{i \mid \hat{x}_i = 0\}$ and set $x_j' = 1$. We need to prove that, (x', y') is never optimal for (MDKP-QUBO).

$$\begin{aligned} & g(x', y') \\ &= \sum_{i=1}^{N} r_i x_i' - \lambda \sum_{d=1}^{D} \left(\sum_{i=1}^{n} w_{id} x_i' - \sum_{t=0}^{M_d} \alpha^t y_{td}' \right)^2 \\ &= \sum_{i=1}^{N} r_i \hat{x}_i + r_j - \lambda \sum_{d=1}^{D} \left(\sum_{i=1}^{N} w_{id} \hat{x}_i + w_{jd} - \sum_{t=0}^{M_d} \alpha_t y_{td}' \right)^2 \\ &\leq g(\hat{x}, \hat{y}) \end{aligned}$$

The last inequality is due to two facts. First,

$$\left(\sum_{i=1}^{N} w_{id} \hat{x}_i + w_{jd} - \sum_{t=0}^{M_d} \alpha_t y_{td}' \right) > 0$$

because previously these constraints were not violated and addition of the new item should violate at least one of the constraints (otherwise $(\hat{x}, \hat{y})$ can not be optimal). Second, as a consequence of our choice of λ, we can infer:

$$\lambda \left(\sum_{i=1}^{N} w_{id} \hat{x}_i + w_{jd} - \sum_{t=0}^{M_d} \alpha_t y_{td}' \right)^2 \geq \lambda \geq R^* \geq r_j \tag{16}$$

So an infeasible solution to (MDKP) can never be optimal for (MDKP-QUBO). Therefore, if $\lambda > R^*$, then x^* should be optimal and feasible for (MKP-QUBO) as well. For the boundary case where $\lambda = R^*$, we reach to the same conclusion, because one can create a feasible solution $(\hat{x}, \hat{y})$ from (x', y'), which means $(\hat{x}, \hat{y})$ is an alternative optimum solution for (MKP-QUBO). Therefore, $g(x', y') = g(\hat{x}, \hat{y}) = r^T \hat{x} \leq r^T x^* = f(x^*)$. $\square$

B. THE 0/1 MULTIPLE KNAPSACK PROBLEM (MUKP)

The 0/1 MUKP involves having multiple knapsacks instead of just one. The objective is to distribute a set of items across several knapsacks, each with its own capacity, to maximize the total value without exceeding the capacity of any knapsack. Formally, given the set $\mathcal{N}$ of items, each with a knapsack dependent revenue r_{ik} and weight w_i, and a set $\mathcal{K}$ of knapsacks with capacity W_k; how can we optimally distribute these items into the knapsacks. For this purpose we declare binary decision variables x_{ik}, which becomes one if item i is placed in knapsack k, and 0 otherwise. Mathematically, the 0/1 MUKP is given by the following formulation.

MUKP:

$$\max f(x) = \sum_{k=1}^{K} \sum_{i=1}^{N} r_{ik} x_{ik} \tag{17}$$

$$\text{s.t.} \sum_{i=1}^{N} w_i x_{ik} \leq W_k, \quad k \in \mathcal{K} \tag{18}$$

$$\sum_{k=1}^{K} x_{ik} \leq 1, \quad i \in \mathcal{N} \tag{19}$$

$$x_{ik} \in \{0, 1\}, \quad i \in \mathcal{N}, k \in \mathcal{K} \tag{20}$$

In this formulation, the objective function (17) maximizes the total value of the selected items. The knapsack capacity constraints (18) limit the total load of each knapsack k to W_k. Next, we introduce a second set of constraints (19) that forbids an item to be placed to more than one knapsack and the formulation ends with binary restrictions (20) on the decision variables.

Similar to MDKP, to develop the QUBO formulation of MUKP, we introduce slack variables $y \in \{0, 1\}^{W_k \times K}$ for constraint set (18) and $s \in \{0, 1\}^N$ for constraint set (19). Then the resulting equations are carried to the objective function by multiplying with penalty parameters λ_1 and λ_2, respectively to form the following penalty function $P(x, y, s)$.

$$\begin{aligned} P(x, y, s) = \lambda_1 & \left[\sum_{k=1}^{K} \left(\sum_{i=1}^{N} w_i x_{ik} - \sum_{t=0}^{M_k - 1} 2^t y_{tk} - \alpha_k y_{M_k k} \right)^2 \right] \\ & + \lambda_2 \left[\sum_{i=1}^{N} \left(\sum_{k=1}^{K} x_{ik} + s_i - 1 \right)^2 \right] \end{aligned} \tag{21}$$

Here again $M_k = \lfloor log_2 W_k \rfloor$ and $\alpha_k = W_k + 1 - 2^{M_k}$), $k \in \mathcal{K}$. Notice that, when the polynomials are expanded, $P(x, y, s)$ will contain a constant term $N\lambda_2$ due to the 1 in the second penalty term. Since the removal of constants has no effect to the optimal solution, from now on we will assume that $P(x, y, s)$ contains only the terms with variables and the constant term is dropped. The resulting QUBO formulation of MUKP is shown below.

MUKP-QUBO:

$$\max \; g(x, y, s) = \sum_{k=1}^{K} \sum_{i=1}^{N} r_{ik} x_{ik} - P(x, y, s) \tag{22}$$

$$x_{ik}, y_{tk}, s_i \in \{0, 1\}, \quad i \in \mathcal{N}, \quad k \in \mathcal{K}, \quad t \in \mathcal{M}_k \tag{23}$$

Theorem 2: For any penalty constant λ_1 *and* λ_2 *such that* $\lambda_1 \geq R^*$ *and* $\lambda_2 \geq R^*$*, where* $R^* = \max\{r_{ik} : i \in \mathcal{N}, k \in \mathcal{K}\}$*, 0/1 the multiple knapsack problem (MUKP) can be reformulated as MUKP-QUBO.*

Proof: Let x^* be an optimal solution for (MUKP). Also let α represent the set of coefficients of y in the binary expansion form. Observe that, x^* is also a feasible solution for (MUKP), then there exists y^*, such that $w^T x^* = \alpha_t^T y^*$ for every $k \in \mathcal{K}$, satisfying constraint set (18). Also, there exists s^* such that constraints (19) are satisfied, resulting in $g(x^*, y^*, s^*) = f(x^*) = r^T x^*$. Let $(\hat{x}, \hat{y}, \hat{s})$ be an optimal solution to (MUKP-QUBO), then $g(\hat{x}, \hat{y}, \hat{s}) \geq g(x^*, y^*, s^*) = f(x^*)$ as (x^*, y^*, s^*) is also a feasible solution to (MUKP-QUBO).

Next we need to show that $g(\hat{x}, \hat{y}, \hat{s}) \leq g(x^*, y^*, s^*) = f(x^*)$, by showing that $\hat{x}$ is a feasible solution to (MUKP). Here we are going to show a contradiction that an infeasible MKP solution x' (but feasible to MUKP-QUBO) can never yield a better objective function value. Let's assume x^* is optimal for (MUKP). Now we will create an infeasible solution x' for (MUKP) and show that $g(x', y', s') \leq g(x^*, y^*, s^*) = f(x^*)$ is never violated.

Let's randomly choose the index pair (j, m) from the compliment of the support of x^*, namely, $(j, m) \in \text{supp}^c(x) = \{(i, k) \mid x^*_{ik} = 0\}$. 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 (MUKP). Let's represent the new (extended) solution as x'. Next, one can construct the following (MUKP-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:

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

Since x^* is an optimal (and feasible) solution, $f(x^*) = g(x^*, y^*, s^*)$. Also, $P^C(x', y', s') = 0$, as it does not contain any terms with x_{jm} and corresponds to already satisfied constraints. Let us define C to represent the sum of the additional terms in the last equation other than $g(x^*, y^*, s^*)$ and $P^C(x', y', s')$. The question is, with our current choices for the values of λ_1 and λ_2, will we always have $C \leq 0$ so that $g(x', y', s') \leq g(x^*, y^*, s^*)$ and so the optimality of the solution (x^*, y^*, s^*) for (MUKP-QUBO) never fails. As x' is infeasible for (MUKP), it has to violate either m-th knapsack capacity constraint or j-th knapsack choice constraint or both. Let us assume that only the first constraint is violated. Then $\sum_{i=1}^{N} w_i x'_{im} - \sum_{t=0}^{M_m} \alpha_t y'_{tm} > 0$ and the minimum magnitude of the violation occurs when $\sum_{t=0}^{M_m} \alpha_t y'_{tm} = W_m$, which implies $y'_{tm} = 1, t = 0, \ldots M_m$. As we assumed the second constraint not to be violated even when $x'_{jm} = 1$, it can only happen if $\sum_{k=1}^{K} x^*_{jk} = 0$ and to offset the penalty $s^*_j = 1$. When $x'_{jm} = 1$, then $\sum_{k=1}^{K} x'_{jk} = 1$ and again we can offset the penalty by setting $s'_j = 0$. Therefore, the term with λ_2 can be removed and the total amount of change in the objective function becomes: $C = r_{jm} - \lambda_1 \left(\sum_{i=1}^{N} w_{im} x'_{im} - W_m \right)$. As we defined $\lambda_1 \geq R^*$ the following relationship can be inferred:

$$\lambda_1 \left(\sum_{i=1}^{N} w_i x'_{im} - W_m \right) \geq \lambda_1 \geq R^* \geq r_{jm} \tag{24}$$

Let us assume the additional item does not cause overflow in the m-th knapsack capacity constraint, but leads to a violation in j-th single-choice constraint. Therefore, we still have $\left(\sum_{i=1}^{N} w_i x'_{im} - W_m \right) \leq 0$ and so there exists a solution y' such that $\sum_{t=0}^{M_m} \alpha_t y'_{tm} = \sum_{i=1}^{N} w_i x'_{im}$ compensates the first penalty term. The amount of violation in the second constraint can only be 1, as this constraint has not been violated as $x^*_{jm} = 0$. Accordingly, $C = r_{jm} - \lambda_2$ and then we can infer that $C \leq 0$ through the below relationship:

$$\lambda_2 \geq R^* \geq r_{jm} \tag{25}$$

When both constraints are violated, we can combine the individual results and construct the following relationship which yields $C < 0$:

$$\lambda_1 \left(\sum_{i=1}^{N} w_i x'_{im} - W_m \right) + \lambda_2 \geq \lambda_1 + \lambda_2 \geq 2R^* > r_{jm} \tag{26}$$

So when $C < 0$, then $g(x^*, y^*, s^*) > g(x', y', s')$, which claims that an infeasible solution to (MUKP) can never be optimal for (MUKP-QUBO), so this case cannot occur. Therefore, if $\lambda_1 > R^*$ and $\lambda_2 > R^*$, then x^* should also be optimal and feasible for (MUKP-QUBO). For the boundary case where $\lambda_1 = R^*$ and $\lambda_2 = R^*$, we obtain the same conclusion above as one can create a feasible solution $(\hat{x}, \hat{y}, \hat{s})$ from (x', y', s'), which means $(\hat{x}, \hat{y}, \hat{s})$ is an alternative optimum solution for (MUKP-QUBO). Therefore, $g(x', y', s') = g(\hat{x}, \hat{y}, \hat{s}) = r^T \hat{x} \leq r^T x^* = f(x^*)$. $\square$

An interesting observation is that, in the third scenario, when both constraints are violated, the requirement of having each penalty parameter being at least R^* is too restrictive. The coefficient of λ_1 can be as large as $w^* = \max\{w_i : i \in N\}$, therefore, as long as $w^*\lambda_1 + \lambda_2 \geq R^*$, both formulations may still have the same optimal solution, which strongly depends on the problem data.

III. SOLUTION METHODOLOGY

To solve the MKP instances created, we use 4 different methods. These are (i) solving the classical integer linear programming formulation of MKP with a solver, (ii) solving the QUBO version of MKP with a solver, (iii) Quantum Annealing and (iv) Quantum Approximate Optimization Algorithm (QAOA). Current state-of-the-art commercial solvers easily solve our instances using advanced operations research methods, which are quite mature and sophisticated and out of the scope of this paper. We will briefly describe the quantum-based methods, namely quantum annealing and QAOA.

A. QUANTUM ANNEALING

Quantum annealing uses the physical properties of quantum systems to address optimization problems [14]. This approach is grounded in a fundamental principle of physics: systems naturally evolve toward their minimum energy state [25]. In a quantum annealer, solving a problem involves designing 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 taking a value of 0 or 1, as determined by the optimal solution [26].

To illustrate the structure of the problem that arises, consider this minuscule $0-1$ knapsack problem instance with two items and a single knapsack: $\max\{3x_1+4x_2 : 2x_1+3x_2 \leq 3, x \in \{0,1\}\}$. Its QUBO version is: $\max\{3x_1 + 4x_2 - \lambda(2x_1 + 3x_2 - y_1 - 2y_2)^2 : x, y \in \{0,1\}\}$. After expanding the squared term, setting $\lambda = R^* = 4$, replacing $3x_1$ and $4x_2$ with $3x_1^2$ and $4x_2^2$ and finally converting the problem into its minimization form, the following final QUBO is formed:

$$\min\{13x_1^2 + 32x_2^2 + 4y_1^2 + 16y_2^2 + 48x_1x_2 - 16x_1y_1 - 32x_1y_2 - 24x_2y_1 - 48x_2y_2 + 16y_1y_2 : x, y \in \{0,1\}\} \tag{27}$$

In a typical application of quantum annealing, each binary variable of the problem is represented as a qubit, and the relationships between variables are mapped onto interactions between qubits, represented as quantum entanglement. The goal is to configure the quantum system so that its lowest energy configuration corresponds to the optimal solution of the QUBO problem. Due to the architecture of quantum computers, the possible number of qubit-to-qubit connections is limited, and therefore additional qubits may be needed to fully map the QUBO onto the quantum hardware.

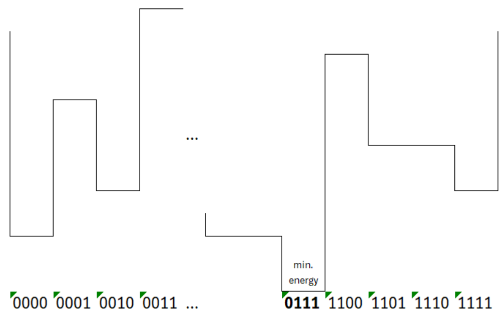

FIGURE 1. The energy landscape of the example QUBO instance.

The annealing process starts with a superposition state of the qubits; however, when the process ends and a measurement is taken, then each qubit falls into a basic state of 0 or 1. Each combination of these basic states has an associated energy level, which corresponds to the objective function of the QUBO in equation (27). When all possible combinations of the basic states are determined and their corresponding energies are measured, one can create the so-called "energy landscape" of the quantum system. Figure 1 represents the energy landscape corresponding to our example and since there are 4 variables in our QUBO example, there are $2^4 = 16$ possible solutions. The height of each state in the energy landscape corresponds to the cost function value for that state. For instance, in the energy landscape, binary strings such as 0011 and 1100 have high energies, which correspond to either not taking any item to the knapsack (0011) or taking both items to the knapsack causing infeasibility (1100). After the initialization of the system by putting all the qubits in superposition, the system is let to evolve, which progressively biases the system toward lower energy states. At the end of the process, the system collapses into a classical state when measured. Each qubit resolves to a binary value (0 or 1). The final configuration with the lowest energy corresponds to the optimal solution of the knapsack problem. For our example, the initial superposition ensures that all 16 possibilities are explored in parallel and the annealing process steers the system toward configurations with lower energies. If the quantum system is set up correctly, then the annealing process is expected to end up at the minimum energy with high probability. In our example, the minimum energy point in the energy landscape corresponds to the optimal solution of 0111, which can be decoded as the solution $x_1 = 0, x_2 = 1, y_1 = y_2 = 1$. Here, the second item is chosen and the auxiliary variables y_1 and y_2 are equal to one as the knapsack is fully occupied by item-2.

B. QUANTUM APPROXIMATE OPTIMIZATION ALGORITHM

The Quantum Approximate Optimization Algorithm (QAOA) is a popular variational quantum algorithm that aims to solve combinatorial optimization problems that are

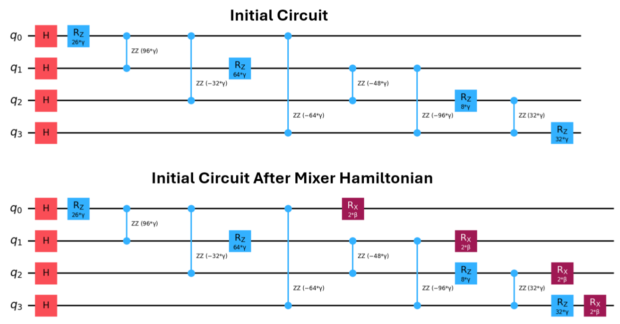

FIGURE 2. **Quantum circuit used in QAOA.**

classically intractable [27]. It belongs to the class of hybrid quantum-classical algorithms and operates on gate-based quantum computers. QAOA uses a sequence of quantum gates to construct a parameterized quantum circuit that encodes the optimization problem. It alternates between two types of gates, corresponding to two Hamiltonians: (i) Cost Hamiltonian Gates, which encode the problem-specific objective function (e.g., a QUBO problem) (ii) Mixer Hamiltonian Gates, that introduce exploration in the solution space. This gate sequence forms one "layer" of QAOA, and the circuit can have multiple such layers [28].

QAOA is designed to run on gate-based quantum computers, such as those built on superconducting qubits (IBM, Rigetti) or trapped ions (IonQ). The algorithm introduces tunable parameters γ (associated with the Cost Hamiltonian) and β (associated with the Mixer Hamiltonian). These parameters are iteratively optimized using a classical non-linear optimizer, based on the outcomes of measurements from the quantum circuit [28].

Figure 2 displays the quantum circuit corresponding to the minuscule 2-item Knapsack instance. As shown in the upper circuit in Figure 2, the problem data is directly mapped to the quantum circuit with certain rotations depending on the coefficients of terms of the QUBO problem instance. The mixer Hamiltonian is then applied, preparing the circuit for the QAOA process. After parameter optimization, a final sampling of the circuit (preferably with a high number of samples) is expected to yield the minimum energy configuration of the system, which should correspond to the optimal solution of the original optimization problem. The implementation of QAOA involves several configurable parameters, including circuit depth, parameter initialization method, and non-linear optimization method, which significantly affect performance and results.

IV. EXPERIMENTAL ANALYSIS

This section explores the impact of penalty parameters used in QUBO formulations on the performance of quantum optimization methods for the MKP. By generating diverse problem instances and varying penalty parameter values, we assess their effects on solution feasibility, quality, and computational efficiency across various quantum technologies, with classical optimization results included for comparison. These experiments also explore how well the theoretical adjustments to the penalty parameters perform in practice for both classical optimization methods and quantum-based methods. By comparing the experimental results across different quantum simulators, we will assess how the characteristics, various settings and the stochasitic nature of each quantum platform affect the performance and scalability of MKP-QUBO solutions.

A. TESTBED

Our testbed contains randomly generated multi-knapsack problems, where the number of items are $N \in \{3, 4, 5, 6\}$ and number of dimensions / knapsacks vary from $D, K \in \{2, 3, 4\}$. The revenue values r are selected as random integers between $[1, 10]$ and the item weights w are chosen as random integers between $[1, 5]$. The knapsack capacities W are again randomly calculated as integers that is between 60% to 80% of the total weight of the items placable to that knapsack to guarantee that not all items are placable at the same time. For the penalty parameters λ, the default value is taken as $\lambda = R^* = \max\{r_{ik}\}$ (or $\max\{r_i\}$). Then a multiplier between 0.5 to 1.5 is used to scale the values of λ, where the multiplier is increased by 0.02, so that 50 different λ values are generated. Our aim is checking how smaller λ values causes infeasibility and also the effect of magnitude of the penalty parameter on the solution time and quality. In total, $4 \times 3 \times 50 = 600$ instances are created for each variant of the multi-knapsack problem. Notice that the problem instances are very small compared to the scales of MKP instances that are efficiently solved by the classical computers. However, as current NISQs contain limited number of qubits, with the current state of quantum technology it is not possible to efficiently solve larger instances.

B. EXPERIMENTAL SETUP AND IMPLEMENTATION DETAILS

To create the instances, we develop a C# console application using Micrsoft Visual Studio Community 2022. We create both the classical MIP formulation and the QUBO formulation of the MKP instances using Gurobi callable libraries and solve both with Gurobi 12.0 Solver [29] with default settings and a maximum solution time of 600 seconds. For the quantum counterparts, we test our instances on three main quantum technologies: Gate based circuits, trapped-ion based circuits and annealing. For the quantum annealing tests we use D-Wave simulator [26]. Using the Python interface of D-Wave, the QUBO corresponding to each instance is used to create the optimization problem and then solved by the annealer. For gate-based experiments, we use IBM Qiskit libraries [30], where the circuits are created from the coefficients of the objective function of the QUBO instance. Then a standard QAOA sub-routine is called to determine the best solution. Last, we use IonQ Python libraries [31] to test our MKP instances under the simulators based on

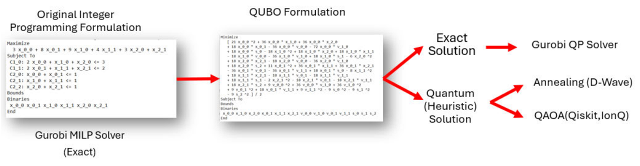
Original Integer Programming Formulation
QUBO Formulation
Gurobi MILP Solver (Exact)
Exact Solution
Gurobi QP Solver
Quantum (Heuristic) Solution
Annealing (D-Wave)
QAOA(Qiskit,IonQ)

FIGURE 3. Solution Methodology.

TABLE 1. Per-cent of instances optimal solution found by the quantum-based method and the average optimality gaps when optimal solution is not found.

N,D	Opt. Solutions Found (%)			Avg. Optimality Gaps (%)		
	Qu. Annl.	QAOA		Qu. Annl.	QAOA	
	D-Wave	Qiskit	IonQ	D-Wave	Qiskit	IonQ
3,2	100%	96.3%	96.3%	0%	1.2%	0.9%
4,2	100%	87.2%	89.7%	0%	2.5%	1.4%
5,2	100%	70.7%	73.2%	0%	3.5%	3.3%
6,2	100%	46.8%	40.3%	0%	6.4%	7.5%
3,3	100%	80.4%	74%	0%	7.8%	7.2%
4,3	100%	47.6%	42%	0%	15.1%	17.5%
5,3	100%	27.2%	20.2%	0%	19.1%	21.5%
6,3	100%	12.5%	6.7%	0%	24.3%	26.8%
3,4	100%	46.7%	43.6%	0%	30.3%	32.1%
4,4	100%	20.5%	12.8%	0%	33.7%	41%
5,4	100%	3.1%	1.2%	0%	51.3%	58.2%
6,4	100%	3.2%	0.4%	0%	55.8%	67.9%

trapped-ion technology. Again, a circuit is created from the coefficient matrix of the corresponding instance and solved by a QAOA algorithm. Figure 3 summarizes the methods used in our experiments.

Each quantum instance is run 10 times with a sample size (or shot) of $1,000$ to reduce the effects of the stochastic nature of quantum mechanics mimiced by the simulators. In both annealing and gate-based solutions, the inherent errors due to the uncertainty of quantum mechanics is thwarted by increasing the number of shots. In each experiment we record, the optimal solution, the optimal objective function value and the solution time. As classical solvers can easily solve the basic integer programming (IP) formulation to optimality, all optimal solutions for all instances are readily available. Therefore, we also record whether the solution of the QUBO version solved by the classical solver and the other three quantum technology-based methods result in a feasible and/or optimal solution or not.

C. COMPUTATIONAL RESULTS FOR MDKP

We first analyze the solution quality of the three quantum methods, which are presented in Table-1. These values are averages of $10 \times 50 = 500$ (10 random instances with 50 different values of penalty parameter) instances for each N and D. Here, the first column shows the number of items and knapsacks of the generated problem instances, respectively. Next three columns show the per cent of the instances (out of 500) the quantum method obtains the optimal solution. The last three columns show the average optimality gaps for the instances when an optimal solution is not achieved by the quantum method. For instance, the first row displays the results for the Multi-dimensional knapsack instance results with 3 items and 2 dimensions. In these instances, D-Wave's annealing method solved all instances to optimality, whereas QAOA based methods of Qiskit and IonQ found the optimal in 96.3% of the instances. The average optimality gap for the remaining 3.7% of the instances are 1.2% and 0.9%, respectively. The optimality gaps are calculated using the formula by $(z_i - z^*)/z^*$, where z_i is the objective function value of the quantum method and z^* is the optimal solution found by the solver. Table-2 displays the average running times of the quantum-based methods recorded. In all instances D-Wave outperforms gate-based methods, where Qiskit simulator is slightly better and faster than IonQ simulator. Observe that Gurobi solver is significantly faster than quantum-based simulators for both the IP and QUBO versions.

TABLE 2. Comparison of running times across different solvers for varying values of N (number of items) and D (number of dimensions).

N,D	Int. Pr. Gur.		Qu. Annl.	QAOA	
	IP	QUBO	D-Wave	Qiskit	IonQ
3,2	0.004	0.026	0.28	6.7	11.2
4,2	0.018	0.043	0.39	14.8	12.3
5,2	0.017	0.065	0.37	17.0	12.8
6,2	0.011	0.145	0.40	92.1	15.2
3,3	0.002	0.022	0.22	11.6	16.1
4,3	0.002	0.040	0.25	18.9	16.9
5,3	0.004	0.035	0.32	23.3	34.5
6,3	0.005	0.087	0.32	35.8	32.2
3,4	0.007	0.033	0.14	23.9	16.8
4,4	0.005	0.079	0.29	36.2	26.2
5,4	0.010	0.049	0.38	54.5	79.7
6,4	0.006	0.036	0.40	91.7	151.8

Table 2 presents the solution time comparison of all the methods.

D. COMPUTATIONAL RESULTS FOR MUKP

Next we present our computational results for the MUKP instances, which are summarized in Table-3 and Table-4. As one can observe, MUKP instances are yielding relatively harder optimization problems compared to MDKP.

As expected, Gurobi solver achieves the fastest solution times for the IP formulation of the MUKP, followed closely by the QUBO version. 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. Observe that for the largest instances, D-Wave may fail to find the optimal solutions, but the optimality gaps are within $1 - 3\%$ of the optimal. In contrast, QAOA-based methods are significantly slower, with solution times increasing as the problem size grows. For the largest instances both gate-based methods fail to provide solutions either with out of memory

TABLE 3. Per-cent of instances optimal solution found by the quantum-based method.

N,K	Opt. Solutions Found (%)			Avg. Optimality Gaps (%)		
	Qu. Annl.	QAOA		Qu. Annl.	QAOA	
	D-Wave	Qiskit	IonQ	D-Wave	Qiskit	IonQ
3,2	100%	39.1%	27.5%	0%	16.8%	19.9%
4,2	100%	16.3%	5.7%	0%	27.9%	31.7%
5,2	100%	3.6%	0.3%	0%	38.1%	42.4%
6,2	100%	0%	0%	0%	42.3%	93.6%
3,3	100%	9.9%	4.9%	0%	45.1%	46.9%
4,3	100%	0%	0%	0%	54%	43.7%
5,3	70.1%	-	-	1.2%	-	-
6,3	56.4%	-	-	2%	-	-
3,4	100%	-	-	0%	-	-
4,4	77.3	-	-	1.1%	-	-
5,4	67%	-	-	1.9%	-	-
6,4	46.2%	-	-	2.7%	-	-

TABLE 4. 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	0.3	23.5	35.1
4,2	0.003	0.14	0.4	30.5	33.8
5,2	0.012	0.22	0.5	184.1	276.8
6,2	0.037	0.34	0.5	183.2	593
3,3	0.004	0.07	0.4	36.6	88.2
4,3	0.005	0.2	0.4	84	437.9
5,3	0.005	1.16	0.5	-	-
6,3	0.004	2.57	0.6	-	-
3,4	0.002	0.13	0.4	-	-
4,4	0.004	0.7	0.5	-	-
5,4	0.004	1.72	0.6	-	-
6,4	0.005	15.61	0.7	-	-

or timeout errors, where again Qiskit simulator slightly outperforming IonQ simulator.

E. EFFECTS OF PENALTY PARAMETER

In the penalty parameter characterizations, we claim that requiring the penalty parameter λ to be at least $\lambda = C^* = \max\{c_{ik}\}$ 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 4 illustrates the average percentage of instances that resulted in feasible solutions under these varying penalty parameter values.

Each data point in the figure represents the average results over $10 \times 12 = 120$ instances (10 random runs across 12 different combinations of N and D/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 IP 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. Observe that, the deterioration of feasibility is much less for MDKP instances compared to MUKP instances. More than 85% of the MDKP instances yield a feasible solution even when the penalty parameter

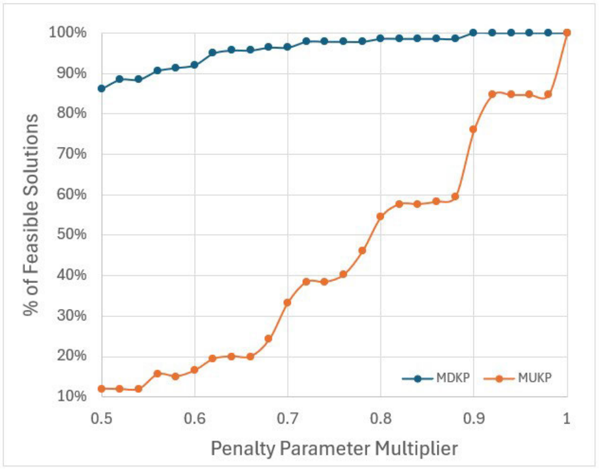

FIGURE 4. Effect of penalty parameter magnitude on solution feasibility.

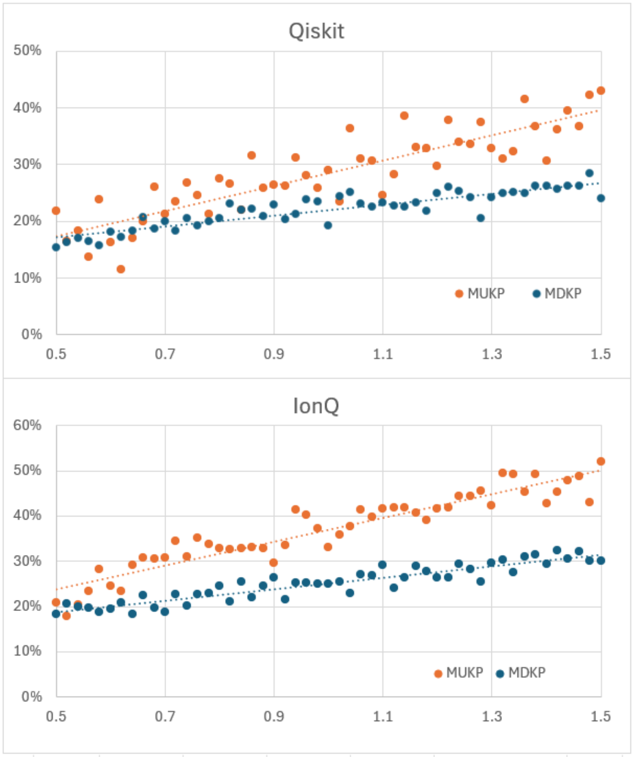

FIGURE 5. Effect of penalty parameter on the average optimality gaps for feasible solutions.

is set as half of the theoretical minimum. Future research could focus on deriving tighter bounds for penalty parameters tailored to specific MDKP / MUKP instances, potentially reducing the need for conservative parameter settings.

Next, we investigate the effect of penalty parameter on the optimality gaps for the feasible solutions. Figure - 5 displays the average optimality gap behavior against the magnitude of the penalty parameter, where the top sub-figure shows the results for Qiskit simulations and the bottom part shows the results for IonQ simulations. As D-Wave finds the optimal

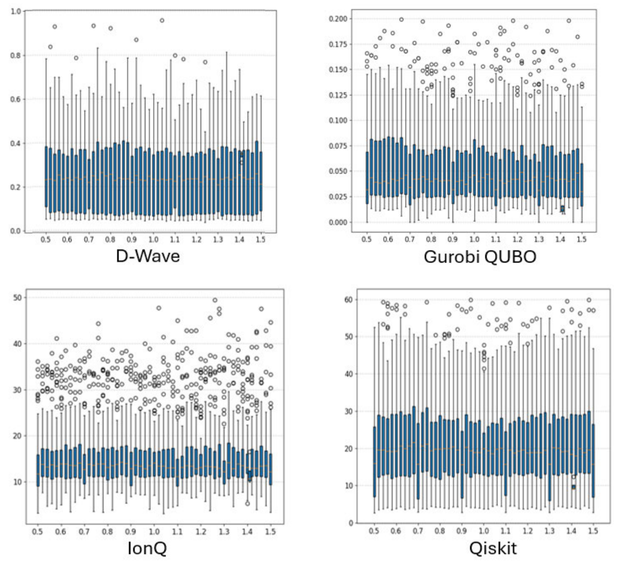

FIGURE 6. Effect of penalty parameter magnitude on running time (MDKP instances).

solutions in more than 99% of the runs, it is excluded from the graphics. The results indicate a strong relation between the average optimality gap and the penalty parameter magnitude. The average gaps are larger for MUKP instances as it is a more complex problem than MDKP. Nevertheless, in both types of MKP variants the average gaps grow as the penalty parameter increases. Therefore having the penalty parameter as small as possible will have a positive effect on the solution quality of MKP instances run on quantum simulators.

Finally, we evaluate how the magnitude of the penalty parameter affects the running time of the solution methods. This analysis is based on the running times for 120 instances across 4 QUBO-based methods. The results are visualized as box-whisker plots for each method. Figure 6 displays the results for MDKP instances, whereas Figure 7 displays the results for MUKP instances. All plots indicate that the magnitude of the penalty parameter has no significant impact on running times.

It is worth noting that these results were obtained using classical simulators of quantum systems, 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 or disprove the observations made here.

F. SOLVING LARGER INSTANCES ON D-WAVE

Our experiments clearly indicate that QAOA-based methods are not scalable even for small-sized MKP instances. In contrast, D-Wave performs well for all small-sized instances. To test the scalability limits for D-Wave annealer, we created larger instances for both MKP variants and executed them on the D-Wave annealer and Gurobi's quadratic solver. We set a time limit of 600 seconds for both solvers and recorded the best solution achieved within this limit. The results displayed in Table 5 are obtained for running each instance 10 times for 50 different penalty parameter multipliers and averaging for only the feasible/optimal solutions.

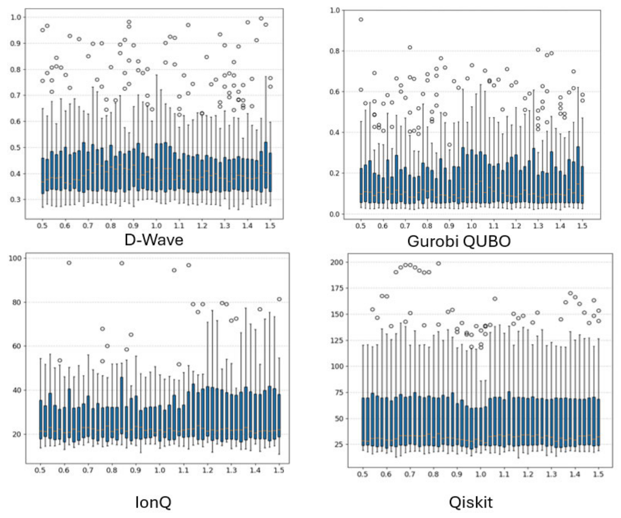

FIGURE 7. Effect of penalty parameter magnitude on running time (MUKP instances).

TABLE 5. Average optimality gaps (%) for larger instances.

MDKP			MUKP		
N,D	D-Wave	Gurobi	N,K	D-Wave	Gurobi
10,5	3.8%	0.0%	10,5	8.1%	1.9%
25,10	21.1%	4.6%	15,5	20.7%	4.6%
50,20	41.3%	51.4%	20,5	27.6%	17.4%
100,25	45.7%	33.6%	25,10	64.9%	10.7%

The Gurobi IP solver can still solve all IP formulations of these larger instances to optimality in less than a second. However, both the D-Wave annealer and Gurobi's quadratic solver struggle with efficiency on larger problems. Both methods fail to find optimal solutions within 600 seconds in most cases, particularly for the largest instances: $(N = 100, D = 25)$ for MDKP and $(N = 25, K = 10)$ for MUKP.

For problem sizes beyond these benchmarks ($N > 100$ for MDKP and $N > 25$ for MUKP), D-Wave fails to find feasible solutions for most instances, so we do not report results for these very large problems. We conclude that within the QUBO domain, the D-Wave annealer presents a promising alternative to classical solvers for problems up to mid-size instances. Nevertheless, the IP formulations of the same instances are solved to optimality substantially faster, demonstrating that quantum-based methods still lag behind their classical counterparts for these problem classes.

G. TESTS ON SIMULATOR SETTINGS AND QAOA PARAMETERS

A critical parameter in QAOA is the number of layers (p) used to create the quantum circuits. As demonstrated by Farhi et al. [28], when $p \to \infty$, the expected energy obtained

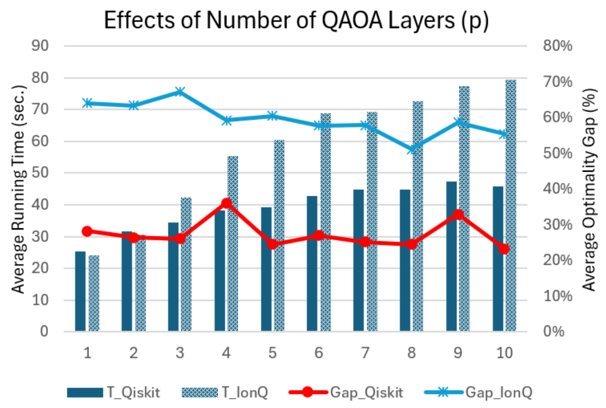

FIGURE 8. Effect of number of layers (p) used in QAOA on running time and optimality gap.

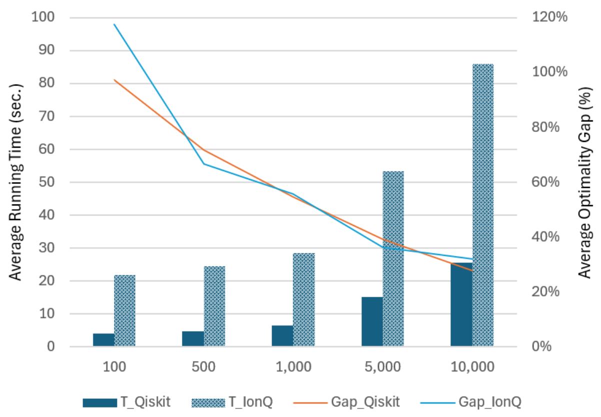

FIGURE 9. Effect of number of shots used in QAOA on running time and optimality gap.

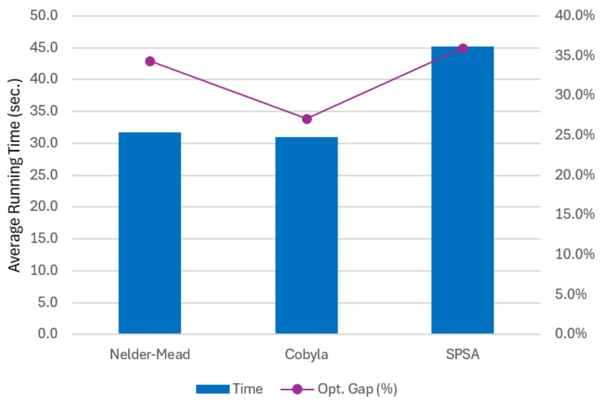

FIGURE 10. Effect of solver choice on QAOA running time and optimality gap.

from the circuit approaches the optimal solution of the underlying optimization problem. Therefore, the choice of p affects the expected value of the QAOA process; however, it also negatively impacts computation time as a larger number of layers results in more complex circuits. Various empirical studies on the choice of p generally favor smaller values [32], [33]. We tested our two QAOA implementations against varying values of $p = 1, \ldots, 10$, with results displayed in Figure 8.

In this figure, the left y-axis values correspond to running times (displayed as bar charts) while the right y-axis values correspond to average optimality gaps (plotted as line charts). We observe that as p increases, the average running times gradually increase, while the average optimality gaps show only modest decreases. The reduction in optimality gap is slightly more pronounced in IonQ simulators compared to Qiskit. Thus, our findings align with previous research [32], [33], suggesting that using smaller p values in circuit construction is preferable unless extended computation time is not a concern.

Another important parameter in QAOA is the number of shots, which refers to the number of times a quantum circuit is executed (or measured) to collect statistical results. Quantum measurements are probabilistic, meaning that running the same quantum circuit multiple times may yield different results due to quantum uncertainty and noise. The number of shots determines how often the quantum computer runs the same circuit to estimate the probability distribution of the measured outcomes. The default value used by Qiskit and IonQ simulator is 1000. We run experiments on sample circuits corresponding to various instances of MKP with 5 different shot values $\{100, 500, 1000, 5000, 10000\}$ to check its effect on both solution quality and running time. The results are displayed in Figure 9.

The results show that number of shots have a significant effect on both running times and optimality gaps. Most real quantum hardware providers charge significant costs as the number of shots increases. Hence, researchers may first test their instances over the free simulators to observe the trade-off between running time and optimality gap before carrying their experiments to actual hardware.

Finally, we analyzed various non-linear solvers used within the QAOA simulators. There are 3 default solvers: COBYLA, SPSA, and Nelder-Mead. Figure 10 displays the average running time and average optimality gaps obtained by each solver. The results indicate that the COBYLA solver performs best in terms of both optimality gaps and speed, followed by Nelder-Mead and SPSA solvers.

V. CONCLUSION

In this work, we applied quantum optimization methods to two variants of the MKP (MDKP and MUKP), using QUBO formulations. Our focus was on characterizing penalty parameters for encoding constraints in QUBO representations and analyzing their impact on solution quality, feasibility, and computational performance. We conducted quantum simulation experiments using both annealing and gate-based QAOA approaches. Our experimental results indicate that:

1) In our application, classical solvers (e.g., Gurobi) remain superior in efficiency and solution accuracy.

Among the quantum approaches, annealing performed promising on small to mid-sized instances, while QAOA struggled with scalability due to circuit complexity and execution time.

2) Penalty parameter calibration affects solution feasibility and accuracy, highlighting the importance of selecting appropriate parameter values to enhance QUBO performance.
3) Both quantum approaches exhibit characteristic strengths and weaknesses. While quantum annealing produces stable results, QAOA is more sensitive to parameter choices, suggesting that improvements in initialization techniques and optimization strategies could enhance its performance.

These insights point to several practical next steps for research:

- While our experiments validate theoretical penalty parameter thresholds, optimizing them to minimize infeasibility and enhance solution quality remains an open challenge. Future work should explore adaptive penalty selection methods that respond dynamically to problem characteristics beyond iterative tuning.
- Our results show that QAOA's performance is dependent on parameter choices, but further investigation is needed to improve stability and scalability. Exploring structured parameter optimization techniques and circuit optimization strategies may enhance its viability.
- Comparing quantum-classical hybrid approaches: Given that neither quantum approach outperformed classical solvers in our study, hybrid strategies should be explored—in both directions. Classical methods could assist with problem size reduction, pre-processing or decomposition [34], while quantum methods may provide inspiration to solve QUBO models faster with accompanying classical methods.
- Evaluating performance on real quantum devices: Finally, while our results establish a baseline in simulation, but testing on actual devices is essential to determine whether quantum methods offer meaningful advantages for optimization in practice. Beyond direct comparisons, such tests could also inform improvements in quantum simulators and circuit optimization techniques.

Overall, our study provides an analytical and experimental foundation for future comparisons between quantum, classical, and hybrid optimization for variants of the MKP, and more generally contributes to the investigation of penalty parameters in quantum optimization from the operations research perspective.

ACKNOWLEDGMENT

Evren Güney acknowledges the support of the Scientific and Technological Research Council of Türkiye (TÜBITAK), which contributed to the development of this research.

REFERENCES

[1] V. Cacchiani, M. Iori, A. Locatelli, and S. Martello, "Knapsack problems—An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems," *Comput. Operations Res.*, vol. 143, Jul. 2022, Art. no. 105693.

[2] W. V. Dam, K. Eldefrawy, N. Genise, and N. Parham, "Quantum optimization heuristics with an application to knapsack problems," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Oct. 2021, pp. 160–170, doi: 10.1109/QCE52317.2021.00033.

[3] A. Lucas, "Ising formulations of many NP problems," *Frontiers Phys.*, vol. 2, p. 5, Feb. 2014.

[4] F. Glover, G. 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, Jul. 2022, doi: 10.1007/s10479-022-04634-2.

[5] A. Verma and M. Lewis, "Penalty and partitioning techniques to improve performance of QUBO solvers," *Discrete Optim.*, vol. 44, May 2022, Art. no. 100594.

[6] M. D. García, M. Ayodele, and A. Moraglio, "Exact and sequential penalty weights in quadratic unconstrained binary optimisation with a digital annealer," in *Proc. Genetic Evol. Comput. Conf. Companion*. New York, NY, USA: Association for Computing Machinery, Jul. 2022, pp. 184–187.

[7] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using QUBO models," 2018, *arXiv:1811.11538*.

[8] T. Pecyna and R. Róüycki, "Improving quantum optimization algorithms by constraint relaxation," *Appl. Sci.*, vol. 14, no. 18, p. 8099, Sep. 2024, doi: 10.3390/app14188099.

[9] C. C. McGeoch, *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*. San Rafael, CA, USA: Morgan & Claypool, 2014.

[10] L. Pusey-Nazzaro and P. Date, "Adiabatic quantum optimization fails to solve the knapsack problem," 2020, *arXiv:2008.07456*.

[11] J. Cai, W. G. Macready, and A. Roy, "A practical heuristic for finding graph minors," 2014, *arXiv:1406.2741*.

[12] A. D. King, T. Lanting, and R. Harris, "Performance of a quantum annealer on range-limited constraint satisfaction problems," 2015, *arXiv:1502.02098*.

[13] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Rev. Modern Phys.*, vol. 90, no. 1, Jan. 2018, Art. no. 015002.

[14] T. Albash and D. A. Lidar, "Demonstration of a scaling advantage for a quantum annealer over simulated annealing," *Phys. Rev. X*, vol. 8, no. 3, Jul. 2018, Art. no. 031016.

[15] F. Arute et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019.

[16] F. Xue, F. Wu, M. Xie, J.-J. Su, and A. H. MacDonald, "Microscopic theory of equilibrium polariton condensates," 2016, *arXiv:1608.06916*.

[17] N. M. Linke, D. Maslov, M. Roetteler, S. Debnath, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Experimental comparison of two quantum computing architectures," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3305–3310, Mar. 2017.

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

[19] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Appl. Phys. Rev.*, vol. 6, no. 2, Jun. 2019, Art. no. 021314.

[20] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, Sep. 2017.

[21] E. Boros and P. L. Hammer, "Pseudo-Boolean optimization," *Discrete Appl. Math.*, vol. 123, nos. 1–3, pp. 155–225, Nov. 2002.

[22] E. Boros, P. L. Hammer, and G. Tavares, "Preprocessing of unconstrained quadratic binary optimization," Dept. Ind. Syst. Eng., Rutgers Univ., New Brunswick, NJ, USA, Tech. Rep., RRR 10-2006, 2006.

[23] R. A. Quintero and L. F. Zuluaga, "Characterizing and benchmarking qubo reformulations of the knapsack problem," Dept. Ind. Syst. Eng., Lehigh Univ., Bethlehem, PA, USA, Tech. Rep. 21T-028, 2021.

[24] A. Awasthi, F. Bär, J. Doetsch, H. Ehm, M. Erdmann, M. Hess, J. Klepsch, P. A. Limacher, A. Luckow, C. Niedermeier, L. Palackal, R. Pfeiffer, P. Ross, H. Safi, J. Schönmeier-Kromer, O. von Sicard, Y. Wenger, K. Wintersperger, and S. Yarkoni, "Quantum computing techniques for multi-knapsack problems," in *Intelligent Computing*, K. Arai, Ed., Berlin, Germany: Springer, 2023, pp. 264–284.

[25] A. Das and B. K. Chakrabarti, "Colloquium: Quantum annealing and analog quantum computation," *Rev. Modern Phys.*, vol. 80, no. 3, pp. 1061–1081, Sep. 2008, doi: 10.1103/RevModPhys.80.1061.

[26] D-Wave. *What is Quantum Annealing?* Accessed: Nov. 11, 2024. [Online]. Available: https://docs.dwavesys.com/docs/

[27] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, "A review on quantum approximate optimization algorithm and its variants," *Phys. Rep.*, vol. 1068, pp. 1–66, Jun. 2024.

[28] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, *arXiv:1411.4028*.

[29] Gurobi Optimization, LLC. (2024). *Gurobi Optimizer Reference Manual*. [Online]. Available: https://www.gurobi.com

[30] I. Quantum. (2023). *Qiskit: An Open-Source Framework for Quantum Computing*. [Online]. Available: https://qiskit.org/

[31] I-IonQ. (2025). *IonQ Quantum Cloud Platform*. [Online]. Available: https://ionq.com/

[32] M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, "Benchmarking the quantum approximate optimization algorithm," *Quantum Inf. Process.*, vol. 19, no. 7, p. 197, Jul. 2020.

[33] M. P. Harrigan et al., "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," *Nature Phys.*, vol. 17, no. 3, pp. 332–336, Feb. 2021.

[34] M. Y. Naghmouchi and W. D. S. Coelho, "Mixed-integer linear programming solver using benders decomposition assisted by a neutral-atom quantum processor," *Phys. Rev. A, Gen. Phys.*, vol. 110, no. 1, Jul. 2024, Art. no. 012434, doi: 10.1103/PhysRevA.110.012434.

EVREN GÜNEY received the B.S., M.S., and Ph.D. degrees in industrial engineering from Boğaziçi University, İstanbul, in 1999, 2002, and 2009, respectively. He is currently an Associate Professor with MEF University, İstanbul. He has published extensively on topics, such as wireless sensor networks and influence maximization in social networks. He is on sabbatical with the University of Applied Sciences and Arts Northwestern Switzerland (FHNW), collaborating with the Paul Scherrer Institute on quantum optimization. His research interests include operations research, integer programming, and combinatorial optimization.

JOACHIM EHRENTHAL received the M.Sc. degree in business administration from Mannheim University and the Ph.D. degree from the University of St. Gallen. Currently, he is a Professor of business information systems with the University of Applied Sciences and Arts Northwestern Switzerland FHNW. He has practical expertise in supply chain IT, transportation, and global trade. He recently completed a sabbatical with Google Zurich, focusing on generative AI agents and agentic systems, and oversees hyperscaler relations as part of the FHNW CIT AI Chapter. His activities at FHNW include teaching generative AI and agentic systems, co-leading the Competence Center Digital Supply Chain, and advancing initiatives in generative AI and quantum technologies.

THOMAS HANNE received the master's degree in economics and computer science and the Ph.D. degree in economics. From 1999 to 2007, he was with the Fraunhofer Institute for Industrial Mathematics (ITWM) as a Senior Scientist. Since then, he has been a Professor of information systems with the University of Applied Sciences and Arts Northwestern Switzerland FHNW and the Head of the Competence Center Systems Engineering, since 2012. He is the author of more than 200 journal articles, conference papers, and other publications. He is an editor of several journals and special issues. His current research interests include computational intelligence, evolutionary algorithms, metaheuristics, optimization, simulation, multicriteria decision analysis, natural language processing, machine learning, systems engineering, software development, logistics, and supply chain management.