IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

QUBO formulations and Quantum Optimization for the Multi-Dimensional Knapsack Problem with Conflict, Forcing and Precedence Constraints

EVREN GUNEY[1], JOACHIM EHRENTHAL[2]

[1]Department of Industrial Engineering, MEF University, Istanbul (e-mail: guneye@mef.edu.tr)
[2]Institute of Business Information Systems, School of Business, University of Applied Sciences and Arts Northwestern Switzerland FHNW (e-mail: joachim.ehrenthal@fhnw.ch)

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

ABSTRACT

Constrained knapsack variants are well-suited for QUBO-based quantum optimization, but adding logical relations can inflate the binary model and complicate penalty selection. This work examines the multi-dimensional knapsack problems augmented with conflict, forcing, and precedence (CFP) constraints using Quadratic Unconstrained Binary Optimization (QUBO) models across classical and quantum platforms. The additional logical constraints are incorporated through quadratic penalty terms that modify existing QUBO coefficients without introducing additional binary variables beyond the slack variables used for the capacity constraints. Computational studies are conducted to assess how each of the three constraint types affects solution behavior, including solution quality, computation time, and sensitivity to penalty parameter magnitudes. The approach is demonstrated on representative small-scale instances using QAOA on IBM Quantum gate-based device and quantum annealing on a D-Wave Advantage processor, with results compared with Gurobi as classical solver. The experiments show that the proposed QUBO formulations achieve optimal or near-optimal solutions for small instances, while showing the sensitivity of current quantum hardware to penalty selection, and scaling. Overall, CFP penalties are variable-neutral relative to the slack-based capacity representation, while the capacity terms dominate logical-qubit count and coupling density. Future work may target more compact or hardware-aware capacity modeling.

INDEX TERMS multi-dimensional knapsack problem, conflict constraints, forcing constraints, precedence constraints, quadratic unconstrained binary optimization, quantum annealing, quantum approximate optimization algorithm, penalty parameters.

I. INTRODUCTION

Quadratic Unconstrained Binary Optimization (QUBO) is a common target form for current quantum optimization platforms, motivating QUBO formulations of constrained combinatorial problems in operations research. For the Multi-Dimensional $0/1$ Knapsack Problem (MDKP), the QUBO model size and interaction structure are largely determined by how the capacity constraints are represented (typically via binary slack variables and the resulting quadratic couplings). In contrast, several practically relevant item-interaction rules, i.e., Conflict, Forcing, and Precedence (CFP) relations, admit compact quadratic penalties that can be added without introducing additional binary variables beyond the slack variables used for capacity modeling.

The Multi-Dimensional Knapsack Problem (MDKP) is a popular extension of the classical Knapsack Problem (KP) where there are N items with revenues $r_i > 0$ and a knapsack with multiple dimensions of quantity D with dimension-dependent 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 revenues of the selected items so that the sum of weights consumed in each dimension does not exceed W_d. Then, the 0/1 MDKP is expressed with the following binary 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 $\mathcal{D} = 1$, 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.

The MDKP 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 optimization approaches, including exact solution methods, heuristics, and metaheuristics, have been highly effective in solving MDKP instances, enabling robust solutions even for 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 (2).

In this work, we focus on natural extensions of MDKP that arise when different interaction requirements exist among the items, specifically three common constraint sets that model CFP type relationships of the items.

The KP with Conflict (C) constraints, sometimes also referred to as disjunctively constraint knapsack problem (3), occur in many real-world applications where not all combinations of items can coexist due to incompatibilities, risks, regulations, or spatial conflicts (4), such as in medical applications, production planning, logistics, portfolio optimization, tournament design, document selection or facility locations with service overlaps (1). Its multi-dimensional (5) and quadratic versions (6) have also been introduced and solved efficiently using exact and heuristic methods.

The KP with Forcing (F) constraints models situations where at least one of two related items must be selected (7). These constraints are sometimes called coverage, inclusion, or dependency rules. The MDKP with Forcing constraints has many real-life applications in various domains, including drug and treatment protocol design, project portfolio selection, sensor network deployment, product line optimization, and academic course planning (1).

The KP with Precedence (P) constraints is another important extension which ensures that selecting an item requires the prior or simultaneous selection of another item with many real-life applications include pit mine optimization, software module installation, task scheduling, supply chain planning and election campaign optimization (8; 9).

Recent advances in quantum computing offer new approaches to combinatorial optimization problems like the MDKP. Quantum computing allows the exploration of solution spaces through different mechanisms compared to classical approaches (10; 11). Many quantum devices support Quadratic Unconstrained Binary Optimization (QUBO) models, making them suitable for exploratory research on problems (12). As scalability is one of the major issues for the current quantum hardware (13), the KP and its variants are well-suited problems due to their simple structures (14; 15; 16).

A key difference between classical optimization techniques and QUBO lies in how constraints are handled. Classical methods explicitly model constraints to ensure feasibility and computational efficiency. In contrast, QUBO integrates constraints directly into the objective function through penalty parameters. Additionally, as the name suggests, QUBO formulations only allow binary variables. When converting classical models into QUBO form, auxiliary binary variables are often required in place of integer or real variables, and slack variables to handle inequalities. This process not only complicates the formulation but also introduces the challenge of constructing a QUBO model that remains computationally efficient. When a general constraint of the form $Ax \leq b$ is directly translated into a QUBO formulation with a penalty parameter P as $P(Ax + s - b)^2$, it introduces extra variables s and results in additional quadratic terms. Designing efficient QUBO formulations has been a longstanding research goal, originating with the work of Boros et al. (17). Subsequent contributions by Lewis and Glover (18), and later Glover et al. (19), proposed several strategies to reduce the size and complexity of QUBO models. For a variety of simple constraints that have only binary variables in it, simpler equivalent penalty terms without introducing any slack variables are suggested in (11) as shown in Table 1.

Classical constraint	Equivalent penalty
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$
$x = y$	$P(x + y - 2xy)$

TABLE 1. Quadratic penalties for common logical constraints in QUBO form.

Further, 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 (20; 21). The characterization and optimization of penalty parameters remain underexplored in

quantum optimization. Boros and Hammer (22) introduced the sum of the coefficients method for pseudo-Boolean optimization problems and later refined it using posiform transformations (17). Basic penalty parameter characterizations for QUBO formulations were provided by Lucas (10) and Glover et al. (11). For KP specifically, Quintero and Zuluaga (14) offer a detailed analysis of penalty parameters, whereas Güney et al. (16) detail theoretical bounds and practical behavior of penalty parameters for the MDKP. A more general approach was proposed by Verma and Lewis (20), who developed a heuristic method to derive lower bounds for penalty parameters in QUBO models. García et al. (21) 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 MDKP by:

- Introducing three constraint extensions that govern relationships of the type Conflicting, Forcing or Precedence.
- Providing efficient QUBO formulations for these variants together with a mathematical characterization of penalty parameters for encoding constraints.
- 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 presents the MDKP variants with CFP constraints. Section III provides the corresponding QUBO formulations, and Section IV derives sufficient penalty-parameter bounds. Section V describes the solution methodology (classical solvers, quantum annealing, and QAOA). Section VI reports the experimental setup and results on simulators and hardware. Finally, Section VII concludes with a discussion of findings and future research directions.

II. MATHEMATICAL MODELS

In this section, we present classical mathematical formulations of the MDKP with side constraints together with their corresponding QUBO formulations. In all models, the sets $\mathcal{N}$ and $\mathcal{D}$ correspond to the items and dimensions with sizes $|\mathcal{N}| = N$ and $|\mathcal{D}| = D$, respectively.

A. MULTI-DIMENSIONAL KNAPSACK PROBLEM WITH CONFLICT CONSTRAINTS

The MDKP with conflict constraints (MDKP-C) can be formulated by adding the conflict constraints. For this purpose we define the conflict set $C \subseteq \{(j,k) \mid j \neq k\}$ that contains the conflict relationships among item pairs, which yields the following binary integer program.

(MDKP-C):

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{N} r_i x_i & & \\ \text{subject to} \quad & \sum_{i=1}^{N} w_{id} x_i \leq W_d & & d = 1, \ldots, D \\ & x_j + x_k \leq 1 & & \forall (j,k) \in C \\ & x_j \in \{0,1\} & & j = 1, \ldots, N \end{aligned}$$

B. MULTI-DIMENSIONAL KNAPSACK PROBLEM WITH FORCING CONSTRAINTS

In the MDKP with forcing constraints (MDKP-F), a set of forcing rules is imposed: the inclusion of one of the two items is forced. Let $F \subseteq \{(j,k) \mid j \neq k\}$ be the set of item pairs that the forcing constraints are defined on, then the following formulation can be derived:

(MDKP-F):

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{N} r_i x_i & & \\ \text{subject to} \quad & \sum_{i=1}^{N} w_{id} x_i \leq W_d & & d \in \mathcal{D} \\ & x_j + x_k \geq 1 & & \forall (j,k) \in F \\ & x_i \in \{0,1\} & & i \in \mathcal{N} \end{aligned}$$

C. MULTI-DIMENSIONAL KNAPSACK PROBLEM WITH PRECEDENCE CONSTRAINTS

This variant includes precedence relationships: some items can only be included if other items are also included.

Let $P \subseteq \{(j,k) \mid x_j = 1 \Rightarrow x_k = 1\}$ represent precedence (i.e., item j can only be selected if item k is selected)

(MDKP-P):

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{N} r_i x_i & & \\ \text{subject to} \quad & \sum_{i=1}^{N} w_{id} x_i \leq W_d & & d \in \mathcal{D} \\ & x_j \leq x_k & & \forall (j,k) \in P \\ & x_i \in \{0,1\} & & i \in \mathcal{N} \end{aligned}$$

III. QUBO FORMULATIONS

The standard QUBO formulation of the MDKP has been analyzed in detail in prior work (16).

The capacity constraints are carried to the objective function as a penalty term with a penalty parameter λ_K. The penalty function $P_K(x,y)$ is constructed by converting the inequalities into equations with the introduction of binary slack variables y_{td} and corresponding coefficients α_{td}, and then squaring the amount of violation of the constraints to obtain:

$$P_K(x,y) = \sum_{d=1}^{D}\left(\sum_{i=1}^{N} w_{id}x_i - \sum_{t=0}^{M_d}\alpha_{td}y_{td}\right)^2 \tag{4}$$

where $M_d = \lfloor log_2 W_d \rfloor$ and $\alpha_{td} = 2^t$ for $t < M_d$ and $\alpha_{td} = W_d + 1 - 2^{M_d}$ for $t = M_d$.

MDKP-QUBO:

$$\max \sum_{i=1}^{N} r_i x_i - \lambda_K P_K(x,y) \tag{5}$$

$$\text{s.t.} x_i, y_{td} \in \{0,1\}, \quad i \in \mathcal{N}, \quad d \in \mathcal{D}, \quad t \in \mathcal{M}_d \tag{6}$$

The additional penalties corresponding to the violation of CFP constraints can be easily constructed. These constraints correspond to the first, second and fourth rows of Table 1, therefore they can be formulated without the need of any slack variables as shown below:

$$P_C(x) = \sum_{(j,k)\in\mathcal{C}} x_j x_k \tag{7}$$

$$P_F(x) = \sum_{(j,k)\in\mathcal{F}} (1 - x_j - x_k + x_j x_k) \tag{8}$$

$$P_P(x) = \sum_{(j,k)\in\mathcal{P}} (x_j - x_j x_k) \tag{9}$$

Observe that, when the polynomial expansion of $P_k(x,y)$ is carried out, it generates (almost) all possible combinations of $x_i x_j$ product terms along with the squared terms x_i^2. Therefore, when the penalty forms of CFP constraints are added to the model, they simply modify the coefficients of existing x_i^2 and $x_i x_j$ terms without increasing the number of QUBO variables. In this regard, while the addition of these constraints leads to a more complex and larger standard Binary Integer Program (BIP), it does not affect the size of the corresponding QUBO formulation. Let's define penalty coefficients λ_C, λ_F and λ_P for the CFP constraints, respectively. The extended QUBO objective function with the addition of all the side constraints then becomes:

$$f(x,y) = r\,x - \lambda_K P_K(x,y) - \lambda_C P_C(x) - \lambda_F P_F(x) - \lambda_P P_P(x) \tag{10}$$

IV. CHARACTERIZATION OF PENALTY PARAMETERS

In this section, we derive sufficient bounds on the penalty coefficients associated with the CPF variants of MDKP to ensure that any optimal solution of the QUBO formulation corresponds to a feasible and optimal solution of the original constrained problem. These results provide theoretical guarantees for penalty selection and serve as a foundation for the computational studies presented later.

Theorem 1. *The MDKP with CFP constraints and its QUBO version yields the same optimal solution when:*

(i) $\lambda_K = \lambda_C \geq R^*$, *where* $R^* = \max\{r_i : i \in \mathcal{N}\}$,

(ii) $\lambda_F \geq \sum_{i\in\mathcal{N}\setminus\{j,k\}} r_i$, *where* (j,k) *is the forcing pair with the two smallest objective function coefficients,*

(iii) $\lambda_P \geq \max\{R^*, \sum_{i\in\mathcal{N}\setminus\{j,k\}} r_i\}$, *where* $(j,k) \in P$.

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 all of the knapsack constraints (2) as well as all of the CFP constraints. 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). Namely, for each type of constraint violation, we construct an infeasible vector x' differing from $\hat{x}$ only in the violating components and show that with the specified penalty values the QUBO objective strictly decreases, contradicting optimality.

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,

$$g(\hat{x},\hat{y}) = g(\hat{x},y^*) = r^T\hat{x} \leq r^T x^* = f(x^*) \tag{11}$$

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).

Case 1: Knapsack constraint violation. The bound for the knapsack penalty parameter λ_K follows directly from the exact-penalty result established in an earlier work (16), where it is shown that $\lambda_K \geq R^*$ guarantees the same optimal solution for both the MDKP and its QUBO formulation. Thus, in the present proof we focus solely on the additional CFP constraints.

Case 2: Conflict constraint violation. Consider a constraint $(i,j) \in C$ with $\hat{x}_i = 0$ and $\hat{x}_j = 1$. Modify the solution by setting $x'_j = 0$. The change in the QUBO objective is

$$g(x',y') - g(\hat{x},\hat{y}) = -r_j + \lambda_C.$$

The decrease in objective function is at most $r_j \leq R^*$; the increase in penalty is exactly λ_C. Thus the change is non-positive whenever $\lambda_C \geq R^*$. Therefore, any infeasible solution with a violated conflict constraint cannot be optimal.

Case 3: Forcing constraint violation. For a forcing pair $(i,j) \in F$ the constraint $x_i + x_j \geq 1$ is violated only when both variables equal zero. Suppose $\hat{x}_i = 0$ and $\hat{x}_j = 1$, and construct an infeasible solution by setting $x'_j = 0$. The worst-case improvement in objective function from freeing capacity is bounded above by

$$\sum_{k\in\mathcal{N}\setminus\{i,j\}} r_k,$$

while the loss from removing item j is r_j. Thus the maximum possible net gain from this modification is

$$g(x', y') - g(\hat{x}, \hat{y}) \leq -r_j - \lambda_F + \sum_{k \in \mathcal{N} \setminus \{i,j\}} r_k.$$

This change is non-positive whenever

$$\lambda_F \geq \sum_{k \in \mathcal{N} \setminus \{i,j\}} r_k,$$

ensuring that any solution violating a forcing constraint is strictly suboptimal. The bound is conservative but sufficient, since the potential improvement from reallocating capacity is overestimated.

Case 4: Precedence constraint violation. For a precedence pair $(i, j) \in P$, the constraint $x_i \leq x_j$ may be violated in two ways.

(a) Setting a forbidden item to one: if $\hat{x}_i = 0$ and $\hat{x}_j = 0$, then setting $x'_i = 1$ yields a change

$$g(x', y') - g(\hat{x}, \hat{y}) = r_i - \lambda_P.$$

This is non-positive whenever $\lambda_P \geq R^* \geq r_i$.

(b) Removing a required predecessor: if $\hat{x}_i = 1$ and $\hat{x}_j = 1$, setting $x'_j = 0$ may free capacity for other items. As in the forcing case, the worst-case net change satisfies

$$g(x', y') - g(\hat{x}, \hat{y}) \leq -r_j - \lambda_P + \sum_{k \in \mathcal{N} \setminus \{i,j\}} r_k.$$

Thus infeasibility cannot improve the objective when

$$\lambda_P \geq \max \left\{ R^*, \sum_{k \in \mathcal{N} \setminus \{i,j\}} r_k \right\}.$$

In all four cases, any infeasible modification of an optimal QUBO solution produces a weakly worse (and typically strictly worse) objective value under the stated penalty bounds. Hence any optimal QUBO solution must correspond to an MDKP-feasible solution, and the two models share the same optimal value and the same optimal set. □

V. SOLUTION METHODOLOGY

To solve our MDKP instances, we use 4 different methods. These are (i) solving the classical integer linear programming formulation of MDKP with a classical solver, (ii) solving the QUBO version of MDKP with a classical solver, (iii) Quantum Annealing and (iv) Quantum Approximate Optimization Algorithm (QAOA). Current state-of-the-art classical solvers easily solve our instances, and are used as oracle in this research. In the following, we will briefly describe the quantum-based methods, namely quantum annealing and QAOA.

A. QUANTUM ANNEALING

Quantum Annealing (QA) is a quantum approach designed to find the ground state (i.e. minimum energy configuration) of an Ising model (23), which many combinatorial optimization problems can be mapped to. In QA, we define a problem Hamiltonian H_p representing the optimization problem, and

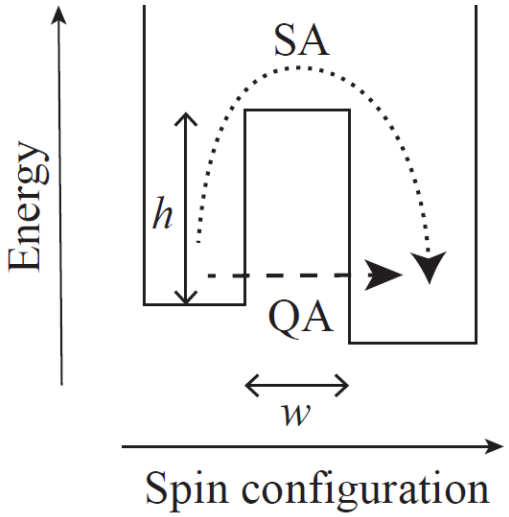

FIGURE 1. Illustration of thermal fluctuation and quantum tunneling in a system with local energy minima separated by an energy barrier (26)

a driver Hamiltonian H_d that is easy to prepare. H_p is usually made of z-Pauli operators (classical terms), while H_d is typically a transverse field built from x-Pauli operators: $H_d = -\sum_j \sigma_j^x$. These two Hamiltonians do not commute (24).

QA works by evolving a time-dependent Hamiltonian:

$$H(t) = f_1(t)H_d + f_2(t)H_p,$$

where $f_1(t)$ and $f_2(t)$ are functions controlling the interpolation between H_d and H_p. The system starts in the ground state of H_d and, if the evolution is slow enough (adiabatic), it ends in the ground state of H_p, which corresponds to the optimal solution of the optimization problem (25; 26).

The success of QA depends on how slowly the Hamiltonian changes. According to the adiabatic theorem:

$$\max \left| \frac{\langle 1(t) | \frac{dH(t)}{dt} | g(t) \rangle}{\Delta(t)^2} \right| \ll 1,$$

where $|g(t)\rangle$ and $|1(t)\rangle$ are the ground and first excited states, and $\Delta(t)$ is the energy gap. Larger $\Delta(t)$ generally means better performance (26).

To highlight the distinctive search mechanism of quantum annealing, we briefly recall the classical simulated annealing (SA) approach (27). SA mimics physical annealing by sampling from a Boltzmann distribution and gradually lowering the temperature to zero. The escape probability over an energy barrier h is: $e^{-h/k_B T}$, where k_B is Boltzmann's constant and T is temperature. If h scales with the system size N, reaching the global minimum may take exponential time.

QA offers a different mechanism: quantum tunneling. Instead of climbing over barriers, QA can tunnel through them as shown in Figure 1 . The tunneling probability is roughly $e^{-\sqrt{hw}/g}$,where g is the strength of quantum fluctuations (related to transverse field), and w is the width of the barrier (28). In optimization terms, the transverse field governs

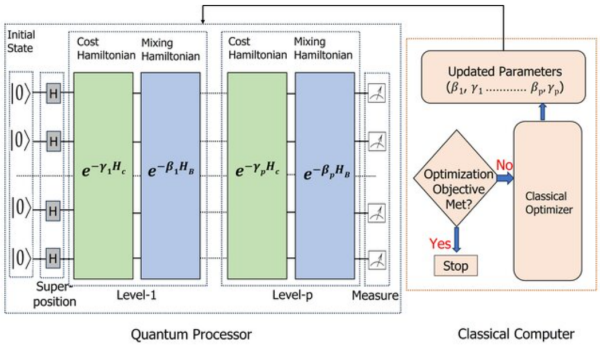

FIGURE 2. Brief outline of QAOA

the intensity of randomized exploration, enabling coordinated variable changes that allow the algorithm to escape narrow local minima. For narrow and tall barriers, tunneling is faster than thermal transitions. This gives QA an advantage over SA, especially for 'glassy' or rugged landscapes which translate into optimization problems with many local optima (29; 30).

B. QUANTUM APPROXIMATE OPTIMIZATION ALGORITHM

The Quantum Approximate Optimization Algorithm (QAOA) solves combinatorial optimization problems using both quantum and classical resources (31). It is designed for gate-based quantum computers and belongs to the class of hybrid algorithms. QAOA works by building a quantum circuit with adjustable parameters. This circuit is made of repeating layers that alternate between two types of quantum operations: (i) cost unitary gates, which represent the optimization objective (such as a QUBO problem), and (ii) mixer unitary gates, which help the algorithm explore different possible solutions. Each pair of these operations forms one layer, and using more layers can help improve the quality of the solution (32). 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 as shown in Figure 2.

VI. EXPERIMENTAL ANALYSIS

In this section, we present our experiments and the results obtained.

A. TESTBED

We construct our testbed by randomly generating MDKP instances, with the number of items $N \in \{4, 5, 6, 7\}$ and number of dimensions $D \in \{2, 3, 4\}$. The objective function coefficients r are selected as random integers between $[1, 10]$, whereas the item weights w are chosen as random integers between $[1, 5]$. The knapsack capacities for each dimension, W_d, are computed as random integers between 60% to 80% of the total weight of the items placable to that knapsack such that not all items are placable at the same time.

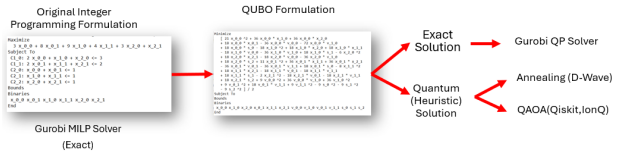

FIGURE 3. Illustration of Solution Approach

For the CFP constraints, we define a density parameter that determines the number of constraints of the given type to be added to the model. Since all these constraints are defined by a couple of variables, where for an N item instance, we can have at most $\binom{N}{2} = N(N-1)/2$ constraints of a given constraint type. We set the constraint density (CD) parameter to be one of $CD \in \{0, 0.1, 0.2, 0.3\}$. For instance, for $N = 5$, we can have at most 10 x_i, x_j combinations and when density parameter is 0.2, we add $0.2 \times 10 = 2$ of that type of constraint to the formulation.

In total, $4(N) \times 3(D) \times 4(CD) = 48$ instances are created for each variant of the MDKP. While these problem instances are compared to what can be efficiently solved using classical method, it reflects current limitations in quantum hardware.

B. EXPERIMENTAL SETUP AND IMPLEMENTATION DETAILS

We use a C# console application developed in Microsoft Visual Studio Community 2022 to create the MDKP instances as well as running all the optimization algorithms. We create both the classical MIP formulation and the QUBO formulations. For the construction of MIPs we use Gurobi callable libraries and then create the corresponding QUBO formulation algorithmically. While constructing the QUBO formulations, for the penalty parameters λ, we use the bounds provided in penalty parameter characterization section. Both the MIP and QUBO are first solved with Gurobi 12.0 Solver (33) with default settings and a maximum solution time of 600 seconds is set as time limit. For the quantum counterparts, we run our instances against three main quantum technologies: Gate based circuits, trapped-ion based circuits and annealing. For the quantum annealing, D-Wave simulator (34) and their annealing algorithm is used, whose Python libraries are available. For gate-based experiments, we use IBM Qiskit Python libraries (35), where the circuits are created from the corresponding QUBO instance. Then a standard QAOA subroutine is called to determine the best solution. Similarly, we call IonQ Python libraries (36) to test our MDKP instances under the simulators based on trapped-ion technology. Again, a circuit is created from the coefficient matrix of the corresponding instance and best solution is obtained by running a QAOA algorithm. The overall solution procedure is summarized in Figure 3.

C. COMPUTATIONAL RESULTS FROM SIMULATORS

In this section, we present the simulation results.

1) Solution Quality Analysis

We first analyze the solution quality of the three quantum methods by showing on what percent of instances the corresponding quantum method reached the optimal solution, as shown in Table-2.

N,D	Conflict			Forcing			Precedence		
	DW	Qi	IQ	DW	Qi	IQ	DW	Qi	IQ
4,2	100	79	64	100	85	39	100	82	55
4,3	100	42	14	100	53	12	100	44	8
4,4	100	17	1	100	21	4	100	20	2
5,2	100	60	26	100	56	14	100	60	25
5,3	100	23	1	100	28	0	100	19	1
5,4	100	7	0	100	7	0	100	4	0
6,2	100	48	17	95	44	7	100	44	11
6,3	100	14	0	100	13	1	100	11	1
6,4	100	1	0	100	3	0	100	4	0
7,2	100	18	3	100	6	0	98	10	3
7,3	100	5	0	94	2	0	100	2	0
7,4	100	0	0	98	1	0	98	0	0

TABLE 2. Percent of instances optimal solution found by the quantum-based method

N,D	Conflict			Forcing			Precedence		
	DW	Qi	IQ	DW	Qi	IQ	DW	Qi	IQ
4,2	0	5	10	0	3	14	0	7	16
4,3	0	18	50	0	26	56	0	37	73
4,4	0	68	111	0	69	129	0	74	117
5,2	0	10	21	0	9	25	0	9	24
5,3	0	32	74	0	39	77	0	39	71
5,4	0	83	155	0	79	153	0	107	165
6,2	0	13	28	≈ 0	17	34	0	19	35
6,3	0	38	75	0	51	92	0	47	79
6,4	0	91	140	0	96	167	0	100	166
7,2	0	36	45	0	117	112	≈ 0	39	50
7,3	0	50	86	≈ 0	145	196	0	58	91
7,4	0	116	165	≈ 0	221	283	≈ 0	119	165

TABLE 3. Average Optimality Gaps (%) for the instances an optimal solution not found

The experimental results from Tables 2 and 3 reveal a performance gap between annealing-based and gate-based simulation approaches. The D-Wave (DW) simulator, which uses annealing, consistently outperforms the gate-based simulators, maintaining an optimal solution success rate of $\approx 100\%$ across nearly all problem instances. Correspondingly, Table 3 shows that the average optimality gap for D-Wave is consistently 0 or ≈ 0, indicating that even when the absolute optimal is missed, the solution remains close to the ground truth. In contrast, the gate-based simulators, i.e. IBM Qiskit (Qi) and IonQ (IQ), show a decline in performance as the problem dimensions (D) increase. This is evidenced by:

- **Scalability Issues**: As D grows, the introduction of more binary slack variables expands the the combinatorial search space, leading to success rates that frequently drop to 0% in Table 2.
- **Growing Optimality Gaps**: Table 3 highlights that as success rates fall, the average optimality gaps for Qi and IQ rise significantly, sometimes exceeding 200% for the most complex instances.
- **Architecture Variance**: Qi generally maintains higher success rates and lower optimality gaps than IQ under the tested settings.

Beyond the solver-level performance differences, examining the impact of the individual CFP constraint types on solution quality reveals additional structure. The data displayed in Table 2 and Table 3 suggests that Forcing constraints present a unique challenge for gate-based optimization. While Conflict and Precedence constraints often allow the "all-zero" state to remain a feasible starting point, Forcing constraints ($x_j + x_k \geq 1$) explicitly forbid it. This requirement forces the quantum optimizer to navigate away from low-weight states, which is reflected in the particularly high optimality gaps for Forcing instances in Table 3.

CD	Conflict			Forcing			Precedence		
	DW	Qi	IQ	DW	Qi	IQ	DW	Qi	IQ
0.1	100	26	11	99	25	7	100	25	9
0.2	100	27	12	99	26	5	100	25	8
0.3	100	24	11	97	28	9	99	25	11

TABLE 4. Percent of instances optimal solution found by the quantum-based method

Table 4 illustrates the impact of increasing constraint density (CD) on the performance of the quantum simulators across CFP variants. The results indicate that the D-Wave (DW) annealing-based simulator remains remarkably stable, maintaining a near-perfect success rate regardless of the density levels. In contrast, the gate-based simulators exhibit low but relatively consistent success rates as density increases from 0.1 to 0.3. While higher density typically implies a more constrained search space that could theoretically aid optimization, the increased number of penalty terms in the QUBO objective appears to offset this benefit for gate-based methods, keeping their success rates stagnant at low levels.

2) Running Time Analysis

We compare the running times of several methods across MDKP instances with increasing problem size. The results are displayed in Figure 4 where the x-axis represents the total number of binary variables in the QUBO model, including slack variables introduced for the multi-dimensional knapsack constraints, meaning the direct comparison with Gurobi Integer Programming (IP) is only approximate (since IP uses its own modeling structure), but the growth trend remains informative.

Qi exhibits a smooth, moderately increasing trend as the number of QUBO variables grows. This is expected because circuit depth and state-vector simulation cost scale exponentially in the number of qubits but remain manageable in this small range. IQ simulator is consistently slower, which we do not investigate further. Both gate-based simulators show runtime growth that is non-linear but still tractable for the tested problem sizes.

DW is consistently the fastest quantum-inspired method, with runtimes around 0.1–0.3 seconds, largely independent

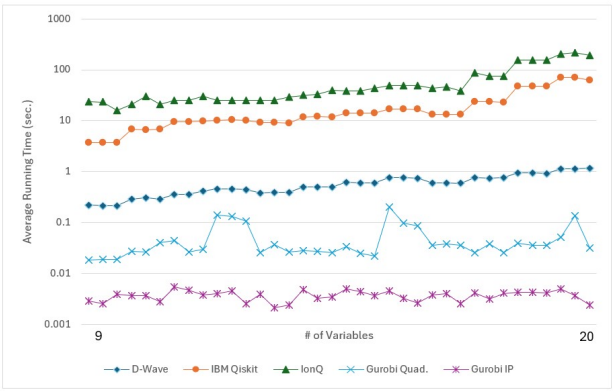

FIGURE 4. Running time comparison of different methods

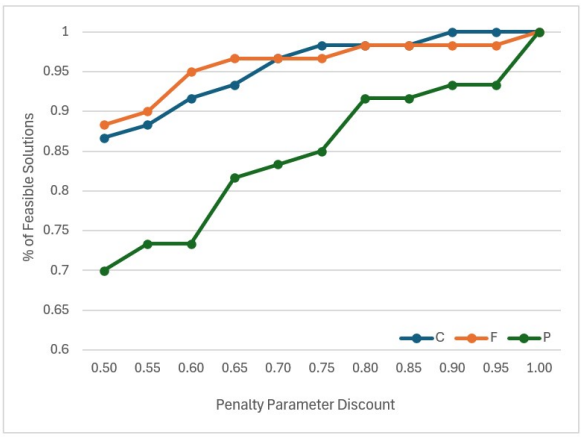

FIGURE 5. Effects of penalty parameter magnitude on solution feasibility

of problem size. The relative flatness of the curve reflects the fact that the simulated annealer does not explicitly scale gate depth or state vector size, and the problem sizes here are small enough that embedding and annealing costs remain stable. This mirrors the typical advantage of annealing-style methods for binary quadratic problems at modest scales.

For the classical solvers we have two baselines: As expected the standard IP solver is extremely fast (milliseconds per run) and shows almost no sensitivity to instance size over this range. This reflects the efficiency of modern solvers on problems of this scale. The quadratic solver is also extremely fast but slightly slower than the IP form. For some instances there are small spikes in the computation time reflecting that solving MDKP classically is still a better choice even with the inclusion of coupling constraints.

3) Effects of Penalty Parameter

Figure 5 illustrates how the penalty parameter discount, i.e. using a fraction of the theoretically sufficient penalty, affects the percentage of feasible solutions produced by the QUBO model for each side-constraint type.

The conflict-only case exhibits a smooth, monotonic improvement in feasibility as the penalty grows. Even at relatively small penalty values ($0.55-0.65$ of the recommended bound), feasibility is already above 90%, and it approaches 100% around $0.90-0.95$. We can conclude that, conflict constraints are easy to enforce with small penalties. Violations are structurally simple (pairwise selection) and the cost Hamiltonian naturally disfavors them as penalties increase. This confirms that the theoretical bound for conflict constraints is close to minimal and not overly conservative.

Forcing constraints show very high feasibility across all discounts, consistently above $93-95\%$, and near-perfect for discounts above 0.7. They are least sensitive to penalty tuning. They represent 'coverage' requirements (select at least one item), and the penalty term tends to align with the natural structure of the optimization objective. In practice, the bound for forcing constraints appears more conservative than necessary, meaning QUBO feasibility is robust even when the penalty is significantly discounted.

The precedence-only model shows the strongest dependence on the penalty parameter. Feasibility starts relatively low (70%) when the penalty is only $50-60\%$ of its theoretical bound, and increases gradually. Near-perfect feasibility is only achieved once penalties reach $0.90-1.00$ of the bound. Precedence constraints are the hardest to enforce in QUBO form and require larger penalties to dominate the underlying objective. Violating a precedence constraint can sometimes increase objective value (e.g., removing a prerequisite with high weight), so insufficient penalties fail to suppress such misleading improvements. This validates that the theoretically derived penalty bound for precedence constraints is tightest and least conservative.

We also check if using extremely large penalty values (up to 100x of the theoretical bound) effect the solution quality. In none of the simulator results, significant change is observed. However, as we will mention in the quantum hardware section, extremely large penalty values result in high degradations in the solution quality.

D. COMPUTATIONAL RESULTS FROM QUANTUM HARDWARE

To complement our simulator studies, we execute a small, non-tuned control study of how our formulations behave on current devices on the D-Wave Leap and the IBM Quantum Platform. We consider 48 QUBO instances spanning $(N, D) \in \{4, 5, 6, 7\} \times \{2, 3, 4\}$ and the four constraint families (Baseline, Conflict, Forcing, Precedence), yielding 12 (N, D) combinations and 4 variants per combination. In this set, the number of logical problem variables N ranges from 4 to 7, while the corresponding QUBO sizes (including slack variables) range from 12 to 23 qubits.

1) IBM Quantum Platform

All hardware runs are executed via IBM Quantum Runtime using the `SamplerV2` primitive within a Runtime `Session`. Following IBM best-practice guidance, all classical steps (LP $\rightarrow$ QUBO/Ising mapping, parameter selection,

transpilation/routing, and simulator validation) are performed offline; the quantum device is used only for final sampling.

We execute 8 representative MDKP-CFP instances on the IBM backend `ibm_kingston` with QAOA depth p=1 and 4,000 measurement shots per instance: four small QUBOs with 12 logical qubits (N=4, D=2; variants Baseline, CFP) and four larger QUBOs with 23 logical qubits (N=7, D=4; the same variants).

For each instance, we compile a backend-optimized, measured, parameter-bound circuit offline, and save it in QPY format for hardware replay. Because backend-aware transpilation produces a *physical* circuit whose register width matches the device (156 qubits on `ibm_kingston`), we apply automatic physical → logical bit remapping using the transpiler layout information before decoding solutions.

On hardware, we sample the saved circuit using `SamplerV2` with dynamical decoupling (XY4) and gate/measurement twirling enabled as default error-suppression settings. From the returned counts we decode each sampled bitstring to the MDKP decision vector x (dropping slack/auxiliary variables) and report the *best feasible* x observed in the sample set, rather than the most frequent full bitstring, which is often infeasible for this constrained problem family on noisy hardware.

For all 8 hardware runs, the best feasible decoded solution matches the Gurobi optimum. However, the probability of sampling the optimum is low and decreases rapidly with problem size: for the N=4, D=2 circuits the empirical probability of sampling an optimal solution is 6.3–6.7% per shot, while for the N=7, D=4 circuits it is 0.65–0.78% per shot.

At a fixed problem size the *type* of CFP constraint has a modest impact on this gate-based sampling difficulty only: across the four variants the optimal-sample probability spans 6.30–6.70% for N=4, D=2 and 0.65–0.77% for N=7, D=4. This is consistent with our formulation: CFP constraints modify linear/quadratic QUBO coefficients but do not change the number of variables, while the dominant circuit complexity is driven by the knapsack capacity encoding (slack qubits and the induced ZZ coupling density), which increases sharply with (N, D).

In terms of timing, runtime metadata shows that wall-clock time is dominated by queueing rather than execution: for these jobs, the runtime-reported QPU usage is 2 seconds per circuit, while queue times range from 8 to 13 minutes.

2) D-Wave Leap

In addition, we conduct an experiment on a D-Wave quantum annealer using the same MDKP-CFP QUBO instances. Here the objective is again to validate the end-to-end workflow QUBO → embedding → annealer → decoded-MDKP, and to compare decoded objective function values and feasibility to classical baselines.

Due to qubit-count and connectivity limitations, experiments on gate-based quantum hardware (IBM Quantum) are restricted to instances up to $N = 7, D = 4$. In contrast, the substantially larger capacity of quantum annealing systems enables evaluation of the proposed QUBO formulation on D-Wave's quantum annealing processor and its classical simulator for instances up to $N = 100, D = 10$.

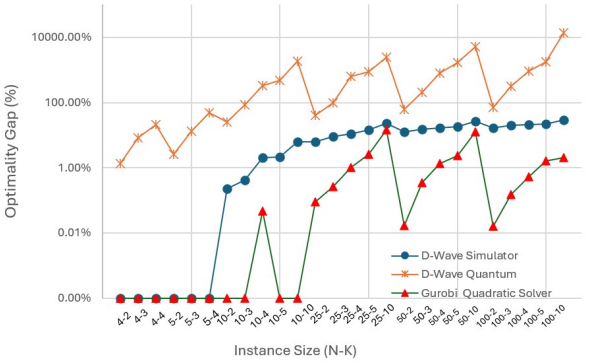

FIGURE 6. Computational Performance Comparison for D-Wave Simulator, D-Wave Quantum Hardware and Gurobi Quadratic Solver for various size of problem instances

We use the D-Wave *Advantage 6.4* backend via the Ocean SDK. Each QUBO instance is converted to an Ising model, minor-embedded onto the hardware graph using the standard heuristic embedding tools, and submitted with *5000* anneals per instance. Chain strengths are set proportional, e.g., $\gamma = 2 \cdot \max_{i,j} |Q_{ij}|$, and default annealing schedules are used.

Figure 6 reports the optimality gap as a function of instance size (N, D) for three solvers on a logarithmic scale. Lower values indicate higher-quality solutions, while increasing gaps reflect either suboptimal or constraint-distorted solutions as problem size grows. Both quantum annealing–based approaches exhibit a clear size-dependent degradation in solution quality as the instance dimension increases. Although the D-Wave simulator outperforms the physical quantum hardware, its optimality gap also grows with problem size, indicating that the penalty-based QUBO formulation becomes increasingly challenging for larger constrained instances. The additional performance gap observed on quantum hardware may be attributable to hardware noise, limited precision in coupler strengths, and minor-embedding overhead, or other aspects which amplify the effect of constraint penalties and chain inconsistencies.

Figure 6 shows the average optimality gaps (%) under both simulated and physical quantum annealing. The observed increase in optimality gap with problem size reflects known limitations of penalty encoding, embedding overhead, and hardware noise, particularly for densely constrained problems. In our setup, the D-Wave simulator consistently outperforms the physical hardware, indicating that performance degradation may be driven both by hardware effects and by the inherent difficulty of scaling penalty-based formulations.

Last, we conduct an analysis on finding out the effects of penalty parameters on the performance of both the D-Wave simulator and quantum processor. Using four fixed MDKP instances ($N = \{4, 5\}, D = \{2, 3\}$), we scale the penalties from 1 to 50 of the theoretical bound. The results

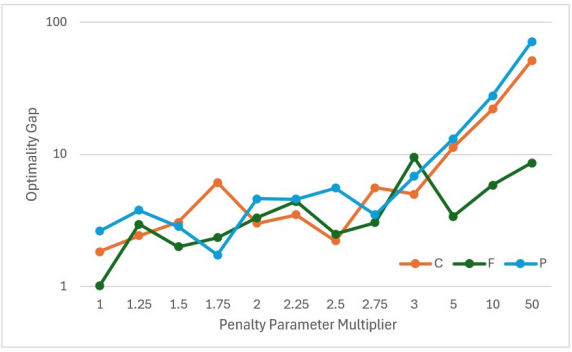

FIGURE 7. **Effects of penalty parameter magnitude on solution feasibility**

are illustrated in Figure 7, which tracks average optimality gaps across all constraint variants. As observed from the figure, excessively large penalties significantly degrade solver performance in practice. Especially beyond 5, all constraint variants (CFP) exhibit a sharp increase in average optimality gaps. This contrast suggests that large penalties amplify hardware-related effects such as limited coefficient precision, embedding distortions, or noise.

When we analyze the effects of penalty scaling separately for each constraint type, clear differences in sensitivity emerge. For the Conflict and Precedence variants, the gaps escalate from near 1 to nearly 100 as the penalty approaches 50. Interestingly, the Forcing constraint variants appear slightly more resilient to extreme penalty scaling compared to the others, though it still follows the overall upward trend. These results suggest that while high penalties guarantee feasibility, they compress the relative energy differences among feasible solutions, making it computationally difficult for the simulators/quantum processors to distinguish the optimal solution from suboptimal but feasible solutions.

VII. CONCLUSION

We presented QUBO formulations for the MDKP with Conflict, Forcing, and Precedence constraints, showing that these relations modify only quadratic coefficients and do not increase the QUBO variable count, and derived sufficient penalty bounds guaranteeing equivalence with the classical formulations.

Across the testbed, the behavior of the quantum methods was driven almost entirely by the size and coupling structure of the knapsack capacity modeling. The CFP constraints altered the QUBO coefficients but had only a minor impact on observed performance in comparison. Differences across quantum approaches reflected platform sensitivity to this structure rather than sensitivity to the specific constraint type.

The results indicate that, under current technical capabilities, the limiting factor for quantum optimization of these MDKP variants is the structure imposed by the knapsack capacity terms. This suggests future work on more compact or structure-aware modeling as well as hybrid approaches that better align problem structure and hardware characteristics.

AUTHOR CONTRIBUTIONS

Conceptualization, methodology, writing: E.G., J.E.; QUBO modeling, penalty-parameter derivations, classical implementation, annealing (D-Wave) and IonQ experiments: E.G.; Quantum computing infrastructure and IBM Quantum experiments: J.E.

ACKNOWLEDGMENT

Evren Güney acknowledges the support of TÜBITAK (The Scientific and Technological Research Council of Türkiye) 2219 Program, which contributed to the development of this research.

REFERENCES

[1] V. Cacchiani, F. Furini, and E. Malaguti, "Knapsack problems — an overview of recent advances. part ii: Multiple, multidimensional, and quadratic knapsack problems," *Computers Operations Research*, vol. 143, p. 105693, 2022.

[2] W. v. Dam, K. Eldefrawy, N. Genise, and N. Parham, "Quantum optimization heuristics with an application to knapsack problems," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2021, pp. 160–170. [Online]. Available: https://doi.org/10.1109/QCE52317.2021.00027

[3] T. Yamada, S. Kataoka, and K. Watanabe, "Heuristic and exact algorithms for the disjunctively constrained knapsack problem," *Information Processing Society of Japan Journal*, vol. 43, pp. 2864–2870, 2002.

[4] U. Pferschy and J. Schauer, "Approximation of knapsack problems with conflict and forcing graphs," *Journal of Combinatorial Optimization*, vol. 33, no. 4, pp. 1300–1323, 2017. [Online]. Available: https://doi.org/10.1007/s10878-016-0035-7

[5] C. Basnet, "Heuristics for the multiple knapsack problem with conflicts," *International Journal of Operational Research*, vol. 32, no. 4, pp. 514–525, 2018.

[6] P. Olivier, A. Lodi, and G. Pesant, "The quadratic multiknapsack problem with conflicts and balance constraints," *INFORMS Journal on Computing*, vol. 33, no. 3, pp. 949–962, 2020. [Online]. Available: https://doi.org/10.1287/ijoc.2020.0945

[7] Y. Takazawa and S. Mizuno, "A 2-approximation algorithm for the minimum knapsack problem with a forcing graph," *Journal of The Operations Research Society of Japan*, vol. 60, pp. 15–23, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:14576649

[8] N. Maiti, P. Pathak, and B. Samanta, "An efficient algorithm for the precedence constraint knapsack problem with reference to large-scale open-pit mining pushback design," *Mining Technology*, vol. 130, no. 1, pp. 8–21, 2021. [Online]. Available: https://doi.org/10.1080/25726668.2020.1866369

[9] E. Güney, "Efficient election campaign optimization using integer programming," *Journal of Industrial En-*

gineering and Management, vol. 11, no. 2, pp. 341–348, 2018.

[10] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. 2, 2014.

[11] F. Glover, G. Kochenberger, R. Hennig, M. Lewis, Z. Lü, H. Wang, and Y. Wang, "Quantum bridge analytics i: a tutorial on formulating and using qubo models," *Annals of Operations Research*, vol. 314, pp. 141–183, 2022. [Online]. Available: https://doi.org/10.1007/s10479-022-04634-2

[12] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, and Y. Wang, "The unconstrained binary quadratic programming problem: a survey," *Journal of Combinatorial Optimization*, vol. 28, pp. 58–81, 2014. [Online]. Available: https://doi.org/10.1007/s10878-014-9734-0

[13] L. Pusey-Nazzaro and P. Date, "Adiabatic quantum optimization fails to solve the knapsack problem," 2020, accessed: 2024-12-04. [Online]. Available: https://arxiv.org/abs/2008.07456

[14] R. A. Quintero and L. F. Zuluaga, "Characterizing and benchmarking qubo reformulations of the knapsack problem," Department of Industrial and Systems Engineering, Lehigh University, Technical Report, 2021, technical Report.

[15] 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. Springer Nature Switzerland, 2023, pp. 264–284.

[16] E. Güney, J. Ehrenthal, and T. Hanne, "Qubo formulations and characterization of penalty parameters for the multi-knapsack problem," *IEEE Access*, vol. 13, pp. 47 086–47 098, 2025.

[17] E. Boros, P. L. Hammer, and G. Tavares, "Preprocessing of unconstrained quadratic binary optimization," Rutgers University, New Jersey, USA, Technical Report RRR 10-2006, 2006, rUTCOR Research Report.

[18] M. Lewis and F. Glover, "Quadratic unconstrained binary optimization problem preprocessing: Theory and empirical analysis," *Networks*, vol. 70, no. 2, pp. 79–97, 2017.

[19] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using qubo models," *arXiv preprint*, vol. arXiv:1811.11538, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1811.11538

[20] A. Verma and M. Lewis, "Penalty and partitioning techniques to improve performance of qubo solvers," *Discrete Optimization*, vol. 44, p. 100594, 2022.

[21] M. D. García, M. Ayodele, and A. Moraglio, "Exact and sequential penalty weights in quadratic unconstrained binary optimisation with a digital annealer," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 184–187.

[22] E. Boros and P. L. Hammer, "Pseudo-boolean optimization," *Discrete Applied Mathematics*, vol. 123, no. 1–3, pp. 155–225, 2002.

[23] A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll, "Quantum annealing: A new method for minimizing multidimensional functions," *Chemical Physics Letters*, vol. 219, no. 5–6, pp. 343–348, 1994.

[24] C. C. McGeoch, *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*. Morgan & Claypool Publishers, 2014.

[25] G. E. Santoro and E. Tosatti, "Optimization using quantum mechanics: quantum annealing through adiabatic evolution," *Journal of Physics A: Mathematical and General*, vol. 39, no. R393, pp. R393–R431, 2006.

[26] A. Rajak, S. Suzuki, A. Dutta, and B. K. Chakrabarti, "Quantum annealing: an overview," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 381, no. 2254, p. 20210417, 2023.

[27] S. Kirkpatrick, C. D. G. Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[28] R. Shankar, *Principles of Quantum Mechanics*, 2nd ed. Boston, MA: Springer, 1994. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4757-0576-8

[29] S. Mukherjee and B. K. Chakrabarti, "Multivariable optimization: Quantum annealing and computation," *The European Physical Journal Special Topics*, vol. 224, no. 1, pp. 17–27, 2015. [Online]. Available: https://doi.org/10.1140/epjst/e2015-02346-0

[30] D. Sherrington and S. Kirkpatrick, "Solvable model of a spin-glass," *Physical Review Letters*, vol. 35, no. 26, pp. 1792–1796, 1975. [Online]. Available: https://doi.org/10.1103/PhysRevLett.35.1792

[31] 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," *Physics Reports*, vol. 1068, pp. 1–66, 2024, a review on Quantum Approximate Optimization Algorithm and its variants.

[32] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014. [Online]. Available: https://arxiv.org/abs/1411.4028

[33] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com

[34] D-Wave. (2022) What is quantum annealing? Accessed: 2024-11-11. [Online]. Available: https://docs.dwavesys.com/docs/

[35] I. Quantum, "Qiskit: An open-source framework for quantum computing," 2023. [Online]. Available: https://qiskit.org/

[36] I. IonQ, "Ionq quantum cloud platform," 2025. [Online]. Available: https://ionq.com/

EVREN GÜNEY received his B.S., M.S., and Ph.D. degrees in Industrial Engineering from Boğaziçi University, Istanbul, in 1999, 2002, and 2009, respectively. He is currently an Associate Professor at MEF University, Istanbul. His research interests encompass operations research, integer programming, and combinatorial optimization. He has published extensively on topics such as wireless sensor networks and influence maximization in social networks. His recent work focuses on quantum optimization.

JOACHIM EHRENTHAL is Professor of Business Information Systems at FHNW. After a sabbatical at Google Cloud, he is now in charge of building out the FHNW School of Business Quantum AI capabilities. Joachim holds a PhD from the University of St.Gallen and an MSc from Mannheim University.