

New QUBO Transformations to Improve Quantum and Simulated Annealing Performance for Quadratic Knapsack

Nicolás Borrajo^{1,2}^a, Juan Marcos Ramírez¹^b, Farzam Nosrati³^c,
Jose Aguilar^{1,6}^d, Vincenzo Mancuso^{1,4}^e and Antonio Fernández Anta^{1,5}^f

¹IMDEA Networks Institute, Madrid, Spain

²ETH Zürich, Switzerland

³Department of Physics, University of Ottawa, Ottawa, Ont. K1N 6N5, Canada

⁴University of Palermo, Palermo, Italy

⁵IMDEA Software Institute, Madrid, Spain

⁶Universidad de Los Andes, Merida, Venezuela

nborrajo@ethz.ch, juan.ramirez@imdea.org, fnosrati@uottawa.ca,
jose.aguilar@imdea.org, vincenzo.mancuso@ieee.org, antonio.fernandez@imdea.org

Keywords: Quantum Annealing, Simulated Annealing, QUBO, Combinatorial Optimization, Quadratic Knapsack.

Abstract: Recent advancements in quantum computing have demonstrated significant potential for solving combinatorial optimization problems, like the quadratic knapsack problem, a constrained binary optimization problem. However, current quantum and quantum-inspired algorithms often require transforming these constrained problems into an unconstrained form, known as Quadratic Unconstrained Binary Optimization (QUBO). Such transformations can significantly impact the algorithms' speed and efficiency. In this study, we evaluate five existing transformation methods and propose four novel approaches. We assess all nine methods using Simulated Annealing and find that three of our approaches outperform existing methods in terms of execution time and the quality and quantity of feasible solutions found. Additionally, we tested these transformations on quantum annealers, which were unable to solve even small problem instances, due to limitations in connectivity and error rates. However, our results highlight the advantages of the new approaches, which reduce the total number of variables in the QUBO representation. This is a critical factor for enhanced performance on emerging quantum hardware, since it also reduces the required number of qubits and the embedding chain lengths.

1 INTRODUCTION

With the emergence of quantum technologies, there has been growing interest in identifying and exploring novel applications for quantum computers. One promising application is solving combinatorial optimization problems, particularly, those represented as *Quadratic Unconstrained Binary Optimization* (QUBO) problems. These problems involve finding the assignment of binary variables that minimizes a quadratic expression formulated using these variables. These advances in quantum optimization the-

oretically enable solving certain NP-hard problems faster than current classical methods (Farhi et al., 2001; Das & Chakrabarti, 2008). However, the practical implementation of quantum optimization methods remains limited due to the low connectivity, high error rate, and limited size of current quantum computers. These limitations restrict their ability to address real-world problems (Rajak et al., 2023).

Among the many combinatorial optimization problems, the *knapsack problem* is a notable example and serves as the focus of this paper. In this problem, a set of items is given, each with an associated value and weight. The goal is to determine the subset of items to include in a knapsack such that the total value is maximized, while ensuring the total weight does not exceed the knapsack's capacity. Beyond its theoretical significance as a foundational problem in combinatorial optimization, the knapsack problem and its

^a <https://orcid.org/0009-0005-4037-7384>

^b <https://orcid.org/0000-0003-0000-1073>

^c <https://orcid.org/0000-0002-0372-4114>

^d <https://orcid.org/0000-0003-4194-6882>

^e <https://orcid.org/0000-0002-4661-381X>

^f <https://orcid.org/0000-0001-6501-2377>

various extensions have numerous practical applications in the real world. For example, it is used in logistics to optimize cargo loading, in finance to build high-performing investment portfolios under budget constraints, and in resource allocation to maximize efficiency with limited resources (Pisinger & Toth, 1998).

Although the knapsack problem can naturally be expressed as a binary optimization problem, because of the weight restriction of the knapsack it is intrinsically constrained. To address this and enable compatibility with current quantum technologies, the problem must be transformed from a constrained binary optimization form into a QUBO formulation.

1.1 Related Work

Recent advances in quantum computing have demonstrated its potential to exponentially speed up the solving of combinatorial optimization problems, sparking considerable interest among researchers. As a result, a growing body of research explores both quantum computers, such as D-Wave’s quantum annealers and circuit-based quantum computers using algorithms like the Quantum Approximate Optimization Algorithm (QAOA) (Farhi et al., 2014), and quantum-inspired algorithms run on classical hardware, such as Simulated Annealing (van Laarhoven & Aarts, 1987) or Fujitsu’s Digital Annealer (Sao et al., 2019), to solve such optimization problems.

The knapsack problem, one of the most extensively studied combinatorial optimization problems, has been the focus of numerous investigations using quantum algorithms. Lucas (Lucas, 2014) presents a transformation of the linear knapsack problem into a QUBO formulation adding W new variables¹, along with similar transformations for other NP-hard problems. This work also introduces an optimization that improves the integration of inequality constraints into the QUBO formulation, requiring only $\log(W)$ additional variables instead of W . This optimization is directly applicable to the knapsack problem.

An evaluation of the two methods proposed in (Lucas, 2014) (using W and $\log(W)$ extra variables) for encoding the knapsack problem into QUBO is provided in (Coffey, 2017). By employing adiabatic quantum computing on small instances of the problem, they investigate the minimum energy gap associated with each encoding. These two QUBO formulations are further analyzed and compared using the QAOA in (Quintero & Zuluaga, 2021), where different reformulations of the QUBO are explored,

with particular attention given to the penalty coefficients required to achieve them. A practical study using D-Wave’s quantum annealers to solve instances of the knapsack problem reveals that current quantum hardware remains insufficient for tackling real-sized problems, especially in comparison to classical solution methods (Pusey-Nazzaro et al., 2020). However, this study only employed the initial QUBO formulation from (Lucas, 2014), leading to suboptimal solutions for smaller problem instances. Other studies comparing this original QUBO encoding with optimized alternative formulations have shown improved results (Bontekoe et al., 2023; Quintero & Zuluaga, 2021), although classical algorithms still outperform all quantum-based approaches.

Other authors have explored variants of the knapsack problem and conducted comparisons with different quantum and classical optimization approaches. In (Van Dam et al., 2021), two optimizations of the QAOA are proposed, with the knapsack problem serving as a baseline to evaluate their effectiveness. Extending this work to a multi-knapsack variant, Awasthi et al. (Awasthi et al., 2023) compare the performance of QAOA with both the Variational Quantum Eigensolver (VQE) and Quantum Annealing methods. Additionally, the multi-dimensional knapsack problem, among other combinatorial optimization problems, has been analyzed using both Digital Annealers and Genetic Algorithms, showing a slight performance gain by the Digital Annealer over the Genetic Algorithms in some instances (Ayodele, 2022).

The *quadratic knapsack problem* is an extension of the original problem in which pairs of selected items can interact and affect the total value (Gallo et al., 1980). The quadratic knapsack problem has also been explored as a benchmark for quantum optimization algorithms. The reformulated QUBO version of this problem was first explored in (Glover et al., 2002), where it was solved using tabu search. Bontekoe et al. (Bontekoe et al., 2023) compare the QUBO formulation of the quadratic knapsack proposed by Glover et al. (Glover et al., 2022) with five newly proposed formulations. This comparison is conducted across various instances of the quadratic knapsack problem, using Simulated Annealing. However, no real quantum hardware was used to compare the different QUBO formulations of quadratic knapsack. Moreover, the proposed QUBO formulations could benefit from further optimizations.

¹ W is the capacity of the knapsack.

1.2 Contributions

This paper focuses on proposing, exploring and comparing different methods for encoding the quadratic knapsack problem into a QUBO formulation. A total of nine distinct QUBO formulations are evaluated and compared. The first two methods are proposed by Glover et al. (Glover et al., 2022) and Lucas (Lucas, 2014), respectively, while the next three are the most efficient approaches presented by Bontekoe et al. (Bontekoe et al., 2023). Additionally, we propose four new QUBOs formulations: three of them are inspired by a formulation from (Bontekoe et al., 2023), while the last one is completely new. Experimental results show increased performance of the four new QUBO formulations compared to the existing methods.

To compare these formulations, two methods will be employed to solve the QUBO problems: *Simulated Annealing* and *Quantum Annealing*. Simulated Annealing is a classical optimization technique that mimics the physical annealing process to probabilistically obtain optimal, or near-optimal, solutions for a given QUBO problem. This was the approach used to test the QUBO formulations in (Bontekoe et al., 2023). On the other hand, Quantum Annealing leverages quantum mechanical properties to solve QUBO problems using specialized quantum hardware known as a quantum annealer. It uses a physical annealing process to find minimum states of binary functions (Finnila et al., 1994).

While Simulated Annealing is capable of handling any QUBO formulation, quantum annealers are limited by their hardware architecture and can only process QUBO problems with specific structures. To enable the use of quantum annealers for arbitrary problem structures, embedding algorithms must be employed to transform the original problem into an equivalent one that is compatible with the annealer’s architecture.

This paper examines the two annealing methods alongside the embedding process for Quantum Annealing, applied to each of the proposed QUBO formulations. To evaluate their performance, a set of knapsack problem instances of varying sizes has been generated. This allows for the observation of how the problem size impacts each formulation, as well as its effect on the annealing methods and the embedding process.

This manuscript is an extended version of the earlier poster paper (Borrajó et al., 2025). The present work provides a deeper exploration of the proposed formulations, together with a more comprehensive analysis of the results obtained using both Simulated

and Quantum Annealing. In particular, this paper places increased emphasis on the embedding process onto the Quantum Annealers, aiming to obtain a better understanding of the causes of the suboptimal performance observed in the Quantum Annealing results.

1.3 Structure of the Rest of the Paper

Section 2 introduces the knapsack problem and its quadratic version. In Section 3, we detail the general methodology for transforming a constrained problem into a QUBO, alongside the definition of the nine QUBO formulations investigated in this study. Section 4 outlines the experimental methodology and provides a discussion of the results. Finally, in Section 5, we present concluding remarks and propose directions for future research.

2 MODEL, NOTATION, AND PROBLEM DEFINITION

This paper investigates the quadratic variant of the knapsack problem, building upon the foundational linear knapsack problem. To provide a clear understanding, we first define the linear knapsack problem. In the context of combinatorial optimization, a linear knapsack problem is typically formalized as consisting of the following parameters.

- $E = \{1, \dots, N\}$ is a set of N items.
- Each item has a weight w_i , $\forall i \in E$.
- Each item has a value v_i , $\forall i \in E$.
- The knapsack has weight capacity W .

All the parameters W , w_i , and v_i are nonnegative integers. Let us define $w_{\max} = \max_{i \in E} w_i$. We assume that $W \geq w_{\max}$, i.e., all items fit in the knapsack. We also assume that $W < \sum_{i \in E} w_i$, i.e., not all items fit at once in the knapsack.

Given these parameters, the objective of the linear knapsack problem is to select a nonempty subset of E to include in the knapsack, which maximizes the total value of the selected items while ensuring that the total weight of the selected items does not exceed the knapsack’s capacity. This problem can be formulated as a Constrained Binary Optimization (CBO) problem by introducing a set of binary variables, one for each item in E , such that,

$$x_i \in \{0, 1\}, \quad \forall i \in E,$$

where x_i is 1 if item i is included in the knapsack, and 0 otherwise. Using these binary variables, we define the knapsack capacity constraint as an inequality

which ensures that the sum of the weights of the included items does not exceed the knapsack's capacity.

$$\sum_{i \in E} w_i x_i \leq W. \quad (1)$$

The last step to complete the optimization problem is to define the actual objective to optimize. For the knapsack problem, the objective is to maximize the sum of the values of the items included in the knapsack. As is common for this type of problem, we define the objective as a minimization, where we simply negate the maximization objective.

$$\min - \sum_{i \in E} v_i x_i.$$

To derive the quadratic version of the knapsack problem from its linear form, additional value coefficients are introduced to account for the inclusion of specific item pairs in the knapsack. These coefficients, denoted as v_{ij} , represent the incremental value added when both items i and j are included in the knapsack. Consequently, the total value contributed by items i and j if both are included in the knapsack becomes $v_i + v_j + v_{ij}$. For simplicity and consistency in the problem formulation, we define $v_{ii} = v_i$, ensuring that the quadratic term for a single item reduces to its linear counterpart (taking advantage of the fact that for any given binary variable x_i , $x_i^2 = x_i$). With this notation, the objective function of the optimization problem can be reformulated as

$$\min - \sum_{i=1}^N \sum_{j=i}^N v_{ij} x_i x_j.$$

Notice that the restriction (1) over the capacity of the knapsack in the quadratic problem remains the same compared to the original linear version.

3 METHODS

3.1 From CBO to QUBO

Most quantum approaches to solving combinatorial optimization problems are designed to handle formulations expressed as QUBO problems, i.e., unconstrained problems. However, many such problems, including the knapsack problem described above, are inherently constrained. To address this, various methods have been developed to transform CBO problems into equivalent QUBO formulations.

These methods incorporate penalty terms into the minimization function to account for each constraint in the original constrained problem. The penalty terms are constructed such that if an assignment of the

binary variables violates a constraint, the corresponding penalty adds a sufficiently large value to the objective function. Conversely, if the constraint is satisfied, the penalty term is evaluated as zero. This design ensures that any non-compliant assignment of variables is rendered suboptimal, effectively preventing it from being the minimum solution of the QUBO formulation.

Therefore, for the given constrained definition of the quadratic knapsack problem, any reformulation into a QUBO problem will take the following form.

$$\min -A \left(\sum_{i=1}^N \sum_{j=i}^N v_{ij} x_i x_j \right) + B \cdot P(X, S), \quad (2)$$

where X represents the vector of binary variables $x_1, x_2, \dots, x_N$, and A and B are positive constants. Here, the original minimization objective of the constrained problem is augmented with a penalty term $P(X, S)$, which is itself a quadratic polynomial over the binary variables X of the problem. Additionally, extra variables S may be introduced in the penalty function to incorporate the original constraints into the QUBO formulation. These extra variables, often referred to as *slack variables*, are denoted by s_k .

In the above equation, A and B are often referred to as penalty constants. These constants are used to ensure that assignments that are unfeasible in the original problem cannot correspond to the minimum of the QUBO expression (2). To enforce this condition, and consistent with the method described in (Lucas, 2014) for defining these constants in the linear knapsack problem, the constants must satisfy the following relation:

$$0 < A \max_{i \in E} \sum_{j \in E} v_{ij} < B.$$

In practice, A is typically set to 1 (as we do). However, the theoretical lower bound for B is often very large, which may lead to suboptimal results when applied to the quantum algorithms discussed in this paper, as explained by Verma and Lewis (Verma & Lewis, 2022). As a result, lower, manually tuned values for B are frequently used instead.

3.2 QUBO Formulations

The nine formulations evaluated in this paper, and their respective source (either taken from the literature or newly defined here), are listed in Table 1 and described below.

Formulation 1 (each_W): The first formulation is based on the original approach presented in (Lucas, 2014), where W binary slack variables are introduced, one for each valid weight of the selected items. The

Table 1: Penalty constants B used for each problem size and QUBO pair. Formulation 1 is defined in (Lucas, 2014), formulation 2 is taken from (Glover et al., 2002), and formulations 3, 4, and 5.1 follow (Bontekoe et al., 2023). The remaining formulations (5.2, 5.3, 5.4, and 6) are newly introduced in this work.

#	QUBO Formulation	Problem Sizes			
		20	35	50	65
1	each_W	10	30	8	8
2	log_W	8	6	6	6
3	each_w_max	8	6	6	6
4	each_w_max_nsr	8	6	6	6
5.1	bontekoe_equality	2	2	4	6
5.2	iterative_equality	2	4	2	2
5.3	half_w_max_equality	2	2	2	2
5.4	sqrt_w_max_equality	4	8	4	4
6	log_w_max	8	4	6	6

penalty function is as follows.

$$P_1(X, S) = \left(1 - \sum_{k=1}^W s_k\right)^2 + \left(\sum_{k=1}^W ks_k - \sum_{i \in E} w_i x_i\right)^2.$$

The first term in the penalty function is zero if and only if exactly one of the W slack variables is set to 1. This determines the weight of the selected items. The second term is zero only when the slack variable set to 1 corresponds to the total weight of the selected items. Both expressions are squared to ensure that any non-zero value becomes positive, thereby preventing it from being the minimum of the penalty function. This guarantees that the penalty is minimized only when the total weight of the selected items matches a valid knapsack weight, which can range from 1 to the maximum capacity W .

Formulation 2 (log_W): Initially presented in (Glover et al., 2022), this approach applies the optimization for inequality constraints defined in (Lucas, 2014). In this formulation, $\lceil \log_2(W+1) \rceil$ slack variables are introduced, replacing the W slack variables used in the previous formulation.

$$P_2(X, S) = \left(W - \sum_{i \in E} w_i x_i - \sum_{k=1}^{\lceil \log_2(W+1) \rceil} 2^{k-1} s_k\right)^2.$$

The newly-defined penalty function evaluates to zero only when the total weight of the selected items, augmented by a slack value derived from the additional variables, equals the maximum knapsack capacity W . Leveraging binary encoding, the slack variables can represent values ranging from 0 to W , enabling them to account for any remaining capacity between the total weight of the selected items and W . This penalty function is currently standard practice for converting inequality constraints into QUBO. This transformation is, indeed, the default method implemented by

most libraries designed to convert constrained problems into QUBO, including those provided by D-Wave and Intel (D-Wave, 2025b).

The following three QUBO formulations have been adopted from (Bontekoe et al., 2023). Specifically, these formulations were identified as the top-performing approaches in this study. Consequently, we include them in our comparisons.

Formulation 3 (each_w_max): This formulation leverages the insight that, for any given knapsack problem, the minimum weight of the optimal solution is $W_{\min} = W - w_{\max} + 1$, where $w_{\max}$ is the largest item weight. This statement holds under the conditions that all item values (and pairwise values, in the quadratic version) are positive, and that not all items can fit in the knapsack simultaneously. If the weight of a given solution is at most $W - w_{\max}$, it is always possible to improve this solution by including any of the remaining items that are not currently in the knapsack, as doing so does not violate the capacity constraint.

This establishes a higher lower bound for the knapsack’s weight, which in turn reduces the total number of slack variables required to formulate the problem as a QUBO problem. In (Bontekoe et al., 2023), this insight was utilized to transform the penalty function of Formulation 1, to use $w_{\max}$ slack variables instead of W .

$$P_3(X, S) = \left(1 - \sum_{k=W_{\min}}^W s_k\right)^2 + \left(\sum_{k=W_{\min}}^W ks_k - \sum_{i \in E} w_i x_i\right)^2.$$

Formulation 4 (each_w_max_nsr): In (Bontekoe et al., 2023), the authors also tested the same formulation but removing the penalty term that enforces the condition of having only one slack variable set to one. In their study, this version produced outcomes comparable to those of the original formulation. The penalty function is as follows,

$$P_4(X, S) = \left(\sum_{k=W_{\min}}^W ks_k - \sum_{i \in E} w_i x_i\right)^2.$$

Formulation 5 (equality): This formulation involves modifying the constraint from an inequality to an equality, enforcing that the total weight of the selected items is equal to the maximum capacity W . This approach eliminates the need for slack variables in the formulation. However, the optimal solution to this formulation may not align with the optimal solution of the original knapsack problem, as the latter may require the knapsack to remain partially empty. Given that the annealing methods used for comparison are probabilistic rather than exact, this approach aims to produce solutions that, even if they violate the

equality constraint, still satisfy the original inequality constraint. As a result, such solutions can approximate the optimal results for the original problem. To better align the solutions with the actual optimal value, a small integer λ can be subtracted from the maximum weight W . Thus, λ serves as a tunable parameter in this formulation, and the penalty function can be written as follows,

$$P_5(X, \lambda) = \left((W - \lambda) - \sum_{i \in E} w_i x_i \right)^2.$$

Four different formulations are defined from this approach by exploring the selection criteria of the parameter λ . The first version is the one presented in (Bontekoe et al., 2023), and referred to as **bontekoe_equality** in this paper. We further propose three modified formulations, as described below.

Formulation 5.1 (bontekoe_equality): This is the original formulation in (Bontekoe et al., 2023), which explores the equality approach. It was only tested for values of λ ranging from 0 to 4, independent of the problem size or parameters.

Formulation 5.2 (iterative_equality): This is the first newly proposed approach, in which values of λ ranging from 0 up to $w_{\max} - 1$ are iteratively tested. Hence, the values of $W - \lambda$ considered include all the possible values that the total weight of the optimal solution could have, as explained in Formulation 3.

Formulation 5.3 (half_w_max_equality) and Formulation 5.4 (sqrt_w_max_equality): The last equality approaches proposed in this paper select only one fixed value of λ for each knapsack problem, corresponding with $\lambda = w_{\max}/2$ and $\lambda = \sqrt{w_{\max}}$, respectively.

Formulation 6 (log_w_max): The fourth new approach proposed is based on the lower-bound used in Formulations 3 and 4 (**each_w_max** and **each_w_max_nsr**), but also applying the optimization used to define Formulation 2 (**log_W**), which reduces the number of required slack variables from $w_{\max}$ to $\lceil \log_2(w_{\max} + 1) \rceil$. The new penalty would then be defined as

$$P_6(X, S) = \left(W - \sum_{i \in E} w_i x_i + \sum_{k=1}^{\lceil \log_2(w_{\max} + 1) \rceil} 2^{k-1} s_k \right)^2.$$

4 PERFORMANCE EVALUATION

4.1 Evaluation Environment

The knapsack problems evaluated in this study were tested using both Simulated Annealing, executed on

a classical computer, and Quantum Annealing, performed on quantum annealers. For both, D-Wave’s Ocean Python library (D-Wave, 2025a) was employed, which provides an implementation of the Simulated Annealing algorithm (executed natively in Python), as well as an API for accessing their quantum annealers. For Quantum Annealing, the tests utilized D-Wave’s Advantage System, a quantum annealer with more than 5000 qubits and more than 35000 couplings (McGeoch & Farré, 2022).

All hyper-parameters used for both the simulated and Quantum Annealing algorithms are left as the default values given by the D-Wave library, as is common for this types of comparisons. For each problem and QUBO formulation pair tested, we perform 5000 Simulated Annealing runs, and 1250 Quantum Annealing runs. For the equality-based formulations that utilize multiple values for the parameter λ (**bontekoe_equality** and **iterative_equality**), the total number of runs was equally divided among the different values of λ , ensuring that the overall number of runs remained consistent with that of the other formulations.

4.2 Penalty Constants

To determine the B constants for the penalty terms across different problem sizes and QUBO formulations, we conducted a search over nine possible penalty values: 2, 4, 6, 8, 10, 15, 20, 25, 30. This search was performed on a small subset of the total problem instances, employing Simulated Annealing with a reduced number of runs. The goal was to evaluate each penalty constant using the Max % metric detailed in Section 4.3, in order to identify the value yielding the best results.

The selected penalty constants for each problem size and QUBO formulation are presented in Table 1. Notably, contrary to the findings in (Bontekoe et al., 2023), our results show that the best penalties differ across QUBO formulations, even when applied to the same set of problems.

4.3 Performance Metrics

For each annealing run, whether simulated or quantum, an assignment to the variables of the knapsack problem is returned. Using this assignment, the total weight and value of the selected items are calculated. Infeasible solutions—those in which the total weight exceeds the capacity of the knapsack for the given problem instance—are discarded. The total value of the feasible solutions is normalized as a percentage of the optimal solution’s value for each problem instance, enabling comparison across differ-

ent instances, each with its unique optimal value.

To compute the optimum value for each problem instance, an exact algorithm developed by Caprara et al. (Caprara et al., 1999) was used. This algorithm efficiently handles small problem instances, including those analyzed in this paper. However, as for any exact solution of an NP-hard problem, its running time does not scale polynomially with the problem size.

With the normalized solutions for each problem, seven metrics are calculated to compare the different QUBO formulations. These include the mean of the normalized values of only the feasible solutions (**Feasible Mean**), the mean of all the values, assuming unfeasible solutions have a value of zero (**Global Mean**), the percentage of feasible solutions found (**Feasible %**), the value of the best solution found (**Max**), the percentage of solutions that achieve the optimal value (**Max %**) and the time taken to compute the solutions in seconds (**Time**).

4.4 Scenarios

A total of 180 quadratic knapsack problem instances were tested in this study. These instances were specifically designed for this research, and include problems with 20, 35, 50, and 65 knapsack items. While instances with a higher number of items were initially tested, the results of the Quantum Annealing experiments and its embedding process revealed significant limitations. The quantum annealers were unable to produce meaningful solutions for larger problem instances, and in some cases, the QUBO transformations of these problems exceeded the capacity of the quantum annealers.

For each of these sizes, 45 distinct problems were generated by varying the knapsack capacity (W) and the maximum item weight ($w_{\max}$), as these parameters directly influence the number of slack variables introduced in the different QUBO formulations evaluated in this paper, which in turn correlate with the effectiveness of each formulation.

The weights and values of the items for each problem instance were randomly selected from a uniform distribution in the range $[1, w_{\max}]$. Additionally, it was ensured that each problem instance included at least one item with a weight equal to $w_{\max}$.

4.5 Simulated Annealing Results

For each problem instance and QUBO formulation pair, 5000 annealing runs were performed, and the metrics described in the previous section were computed. Not surprisingly, the **each_W** formulation performs the worst among all approaches, yielding only

a few feasible solutions even for the smallest-sized problems (1% for problems with 20 items and zero for bigger problems) and exhibiting considerably longer execution times. This QUBO formulation is practically unusable for any reasonably-sized problem, which was expected given that it is the unoptimized initial formulation, requiring the largest number of variables. For this reason, its metrics are not even presented in the rest of the paper.

For the rest of the formulations, the results are summarized in Figure 1, where for each metric, the values of each QUBO formulation and problems of the same size are displayed as a box plot. The results show that the equality approaches generally outperform inequality formulations across all evaluated metrics. The absence of slack variables, combined with a simpler QUBO formulation, appears to increase the likelihood of finding feasible solutions for the equality QUBO as shown in Figure 1.c (note that the Feasible % for the **half_w_max_equality** is almost always 100%, and therefore the boxes for this formulation are not visible). Additionally, as the tested values of $W - \lambda$ are typically close to the knapsack optimum in most cases, the equality formulations yield superior overall performance in all metrics compared to the inequality methods (see Figure 1.e). The reduced number of total variables not only simplifies the problem but also decreases experimental runtime, further widening the performance gap between the approaches as seen in Figure 1.f.

The importance of selecting appropriate values for λ becomes evident when comparing various equality methods. The **bontekoe_equality** formulation, with λ values ranging from 0 to 4, demonstrates strong performance for problems with 20 items. In these cases, the optimum value often falls within the tested $W - \lambda$ range (see Figure 1.e). However, as problem sizes grow, this formulation's performance deteriorates because the optimum for larger problems frequently lies outside this range. Additionally, finding feasible solutions with such high total weights becomes increasingly harder, which can be observed in Figure 1.c.

In contrast, the **iterative_equality** method, which tests all reasonable values of λ , achieves the most stable and promising results across all problem sizes among the equality formulations. This stability, however, comes at the cost of slightly lower average performance for some problems, as many of the tested λ values are suboptimal. The **half_w_max_equality** and **sqrt_w_max_equality** methods, on the other hand, evaluate only a single λ value for each problem, derived relative to $w_{\max}$. This approach results in greater variability in performance, as these methods excel only for problems where the optimum lies near the

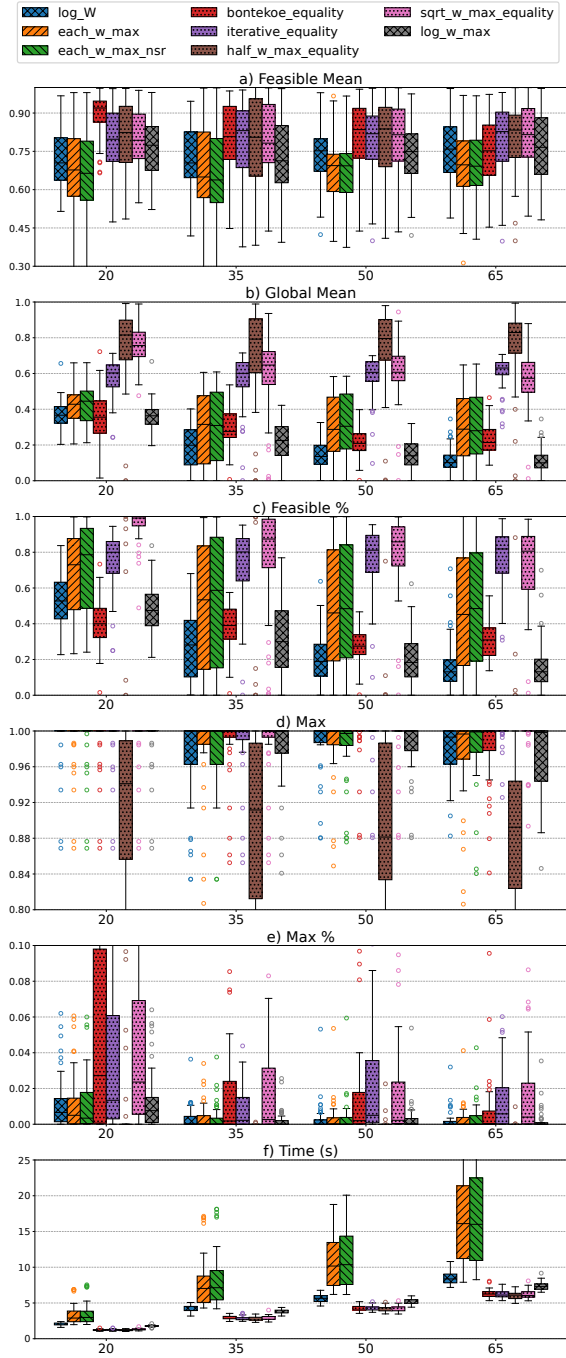


Figure 1: Simulated Annealing results for the different QUBO formulations. Each box shows a metric of a given QUBO (color) and different problem sizes (x-axis). Some plots have been truncated vertically for visibility.

selected $W - \lambda$ value. For this reason, **half_w_max_equality** obtained low values for the Max % metric (see Figure 1.e, where the boxes for **half_w_max_equality** are at or close to 0), as the optimum value frequently exceeded $W - w_{\max}/2$. Nevertheless, for

the same reason, this formulation excelled in finding decent feasible solutions (see Figures 1.b–c), since $W - w_{\max}/2$ remains well below the knapsack’s capacity limit while still being high enough to yield good solutions. The value of $\lambda = \sqrt{w_{\max}}$ in the **sqrt_w_max_equality** formulation better approximates the optimal value, resulting in some of the best average metrics among all QUBO formulations. However, it is important to note that this observed advantage may not necessarily generalize to all problem instances or knapsack sizes.

Among the other four inequality-based formulations, **each_w_max** and **each_w_max_nsr** show nearly identical performance, consistent with the findings of (Bontekoe et al., 2023). The last two formulations, **log_W** and **log_w_max**, which incorporate logarithmic optimization to reduce the number of slack variables, also exhibit similar results. Between these two groups of formulations, the logarithmic ones demonstrate slightly better average and optimal solution metrics (see Figures 1.a and 1.e), albeit with a lower feasibility percentage (see Figure 1.c). However, due to the significant reduction in the total number of variables, these latter methods execute much faster (see Figure 1.f). Among the logarithmic formulations, **log_w_max** slightly outperforms **log_W**, as it employs fewer variables overall.

4.6 Problem Embeddings

To solve a QUBO problem on a quantum annealer, each variable is represented by a qubit, while the quadratic terms in the QUBO are encoded using couplers between the two corresponding qubits. However, current quantum annealers feature sparse architectures, where each qubit can interact only with a limited number of neighboring qubits. In contrast, knapsack QUBO formulations are inherently dense, as they involve quadratic terms between every pair of variables. Consequently, a direct mapping of a knapsack QUBO onto the architecture of a quantum annealer is infeasible because it would require a fully connected architecture in which every qubit is coupled with all others.

To solve this problem, an embedding is constructed between the original QUBO formulation and the architecture of the quantum annealer. This embedding maps each QUBO variable to a chain of one or more physical qubits that are strongly coupled, ensuring that they collectively represent the same value. These chains can then interact with other chains via any of their constituent qubits, enabling the embedding of the original densely connected QUBO graph onto the sparse architecture of the quantum annealer.

Many algorithms exist to generate these embeddings. D-Wave offers two such algorithms to embed problems onto their quantum annealers. The first and default algorithm, *minorminer*, is a heuristic-based algorithm capable of embedding any input graph onto the target architecture (Cai et al., 2014). The second, *busclique*, is specifically designed to generate embeddings for fully connected (or clique) graphs in polynomial time (Boothby et al., 2016). Compared to minorminer, busclique typically produces better embeddings for such graphs in a fraction of the time. Since all the QUBO formulations discussed in this paper produce fully connected graphs, the busclique algorithm was chosen to generate the problem embeddings.

Since the embedding relies solely on the graph structure of the problem and all QUBO formulations share the same fully-connected structure, QUBOs with an identical number of variables can utilize the same embedding. This significantly reduces the number of embeddings that need to be computed, as many problem instances share the same variable count. Moreover, certain QUBO formulations, such as **each_w_max** and **each_w_max_nsr**, or equality-based algorithms, produce the same number of variables for any given problem, further minimizing the need for distinct embeddings.

The total time required to generate all problem embeddings was approximately 5 minutes. Given the relatively small size of the problems and the variability of the performance of a classical computer, no significant difference in execution time was observed between different formulations and problem sizes. In Figure 2, the initial number of variables in the QUBO (a), the total number of qubits used after the embedding (b), and the average qubit length of the embedding chains (c) are shown for each problem size and QUBO approach (grouping the ones that produce the same embedding).

These results underscore the importance of minimizing the number of variables in a QUBO formulation. As the number of variables increases, the embedding process not only requires more time but also results in embeddings of lower quality. Specifically, these embeddings demand more total qubits and produce longer qubit chains, which increase the likelihood of errors when solving the problem on the quantum annealer.

The equality-based embeddings, being that they do not introduce any additional slack variables to encode the QUBO, have the least embedding overhead cost (see Figures 2.b–c). The results also show a definitive advantage between the QUBO formulations that have a logarithmic reduction in the num-



Figure 2: Embedding Metrics for the different equivalent QUBO formulations.

ber of slack variables, and those that don't, which provides an additional benefit to these formulations when leveraging quantum annealers, compared to the results obtained using Simulated Annealing.

Additionally, since the number of qubits in a quantum annealer is limited, some larger instances of QUBOs generated using the **each_W** and **each_w_max** formulations resulted in embeddings that exceeded the available qubit capacity of the quantum annealer. This limitation prevented these instances from being solved in such machines.

4.7 Quantum Annealing Results

Once the embeddings for the different QUBO formulations have been calculated, the problems can be solved on the quantum annealers. Due to the limited QPU time available on the D-Wave Quantum Annealing platform, a reduced number of 1250 runs per problem was selected. Additionally, only four of the initial nine QUBO formulations were chosen for testing on the quantum annealer, based on the Simulated Annealing results presented earlier.

The four selected QUBOs are **log_W**, **each_w_max**, **sqrt_w_max_equality** and **log_w_max**. The **each_W** formulation was excluded because of its sub-optimal results in Simulated Annealing. Similarly, **each_w_max** was chosen over **each_w_max_nsr**, as both are very similar in both formulation and performance in Simulated Annealing, but **each_w_max** is

formally more complete, and the additional restriction with respect to **each_w_max_nsr** does not significantly impact the complexity or size of the QUBO, making **each_w_max** the most appropriate choice.

Among the equality-based formulations, **bon-tekoe_equality** was excluded due to its inability to scale effectively to larger problems. Between **half_w_max_equality** and **sqrt_w_max_equality**, given they have conceptually the same QUBO formulation, the latter was selected as it achieved better metrics in Simulated Annealing, particularly in identifying optimal solutions. Although the **iterative_equality** method exhibited strong performance and was the most stable among the equality-based methods, it was not selected for quantum annealer testing. This decision was driven by the significant initialization overhead associated with the annealer’s architecture. The iterative approach, which involves solving numerous small problems with a limited number of runs each, is unsuitable given the restricted QPU time available.

Due to the limited number of total runs performed, the noise in the results is significant. Nevertheless, these results provide valuable insights. As anticipated, performance degrades significantly with increasing problem sizes. Due to the full connectivity of the QUBO graphs generated for the knapsack problem, as discussed in the previous section, the embeddings needed to program the problems into the annealer’s architecture increase considerably the total number of qubits needed, and both the number of qubits and the chain lengths of the embeddings worsen with problem size (see Figure 2). Current quantum annealers are highly sensitive to environmental noise, therefore longer chain lengths increase the likelihood that any of the qubits in the chain will be affected by noise, which in turn, can break the chain and lead to suboptimal results.

This effect is evident in the results obtained. For larger knapsack problem instances of 50 and 65 items, the quantum annealer was nearly incapable of generating feasible solutions (1.5% and 0.1% respectively, among all formulations), and those few found by only some of the QUBO formulations were far from optimal (0.05 and 0.1 Feasible Mean, respectively). For this reason, we only show the results of problems with 20 and 35 items in Figure 3. Even for these smaller problem instances, while the number of feasible solutions was more substantial (see Figure 3.c), their quality was still suboptimal (see Figures 3.a–b). In the case of problems with 20 items, the results were even comparable to those obtained by randomly sampling the solution space, whose metrics are shown in Figure 3 as the **random** approach. However, for problems with 35 items, the quantum annealer did find

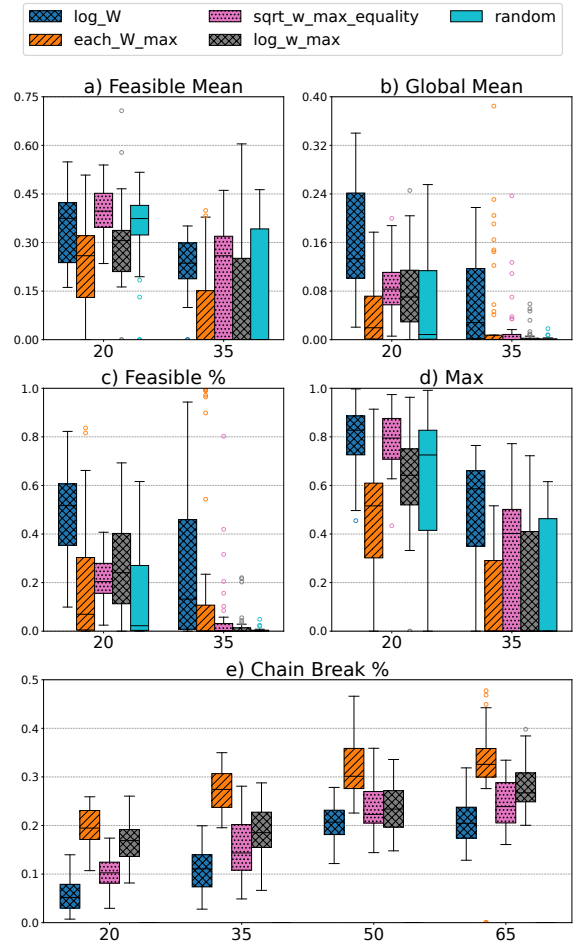


Figure 3: Quantum Annealing results for the chosen QUBO formulations. Each box shows a metric of a given QUBO (color) and different problem sizes (x-axis). The results of the random sampler (light blue) are also shown for comparison.

significantly more feasible solutions than the random sampler (see Figure 3.c), which suggests that noise interference in quantum annealers does not entirely randomize their results, at least for problems of up to this size.

Figure 3.e shows the percentage of chains created by the embedding process that are broken, which further explains the poor performance observed and highlights the effect of noise. As explained previously, the length of the chains in the embeddings increases with the number of variables, which, as shown in the Figure 3.e, has a direct impact on the probability of chain breaking. The Figure also indicates that the formulations achieving the best results on the quantum annealer are those with the lowest chain-break percentage.

Given that the feasible solutions identified by the quantum annealer are, on average, significantly dis-

tant from the optimal (see Figure 3.a), and the best solution found is also generally far from the optimal (see Figure 3.d), the Max % metric was omitted from Figure 3. Similarly, the Time is not shown on Figure 3 as the quantum annealing time for each QUBO formulation and problem size pair is very similar. This is due to the substantial initialization overhead previously mentioned, which makes the annealing time almost invariant to the amount of qubits used. In total, two minutes of quantum annealing time were allocated for the experiments, evenly distributed across QUBO formulations and problem sizes.

Further analysis of the results in Figure 3 show significant differences when compared to the outcomes from Simulated Annealing. Notably, the **log-W** formulation consistently produced the best results, while the performance of the other formulations varied depending on the problem size. In contrast to the trends observed in Simulated Annealing, no clear pattern emerged for the quantum annealer results. Factors such as high noise levels, suboptimal performance, and the inability to scale up to address real-sized problems, led us to conclude that none of the tested formulations consistently outperformed the others. Additionally, the hypothesis that reducing the total number of QUBO variables significantly enhances annealing performance could not be proven, as the knapsack problems for which the difference in the number of variables between QUBO formulations is significant yield almost no feasible solutions, and therefore, could not be compared.

5 Conclusions and Future Work

The experiments conducted using Simulated Annealing highlight the critical importance of minimizing the total number of variables when translating the quadratic knapsack problem into a QUBO formulation. The new methods introduced in this work outperform previously defined QUBO formulations, particularly the equality-based transformations, for which the choice of an optimal value for the parameter λ was shown to be crucial for the performance of these formulations.

The quantum annealer was unable to solve even relatively small instances of the quadratic knapsack problem. This limitation is likely attributed to the inherent noise in the machine, which had a pronounced effect on the long qubit chains required to embed the QUBO formulations into the quantum annealer's architecture. These long chains are necessary due to the intrinsic full connectivity of the QUBO transformation for the quadratic knapsack problem.

Despite these poor results, the embedding process underscored the increased importance of reducing the total number of variables in QUBO formulations. This reduction leads to fewer qubits and shorter chain lengths, which appear to be key constraints for the performance of current quantum annealers. As quantum hardware technology advances, improvements in qubit connectivity and reductions in error rates will enable the solution of larger problems with better performance. In this context, the significance of efficient embedding strategies and QUBO formulations will become increasingly evident.

Given the exceptional performance of equality-based algorithms compared to other QUBO formulations and the significant variation observed in equality formulations depending on the chosen value of λ , further research must investigate alternative methods for selecting this parameter. While the iterative approach appears to yield the most stable results, uniformly sampling the parameter space may not be the optimal strategy. A targeted approach that allocates more annealing runs to promising values of λ could potentially enhance outcomes.

As advancements in Quantum Annealing hardware continue, alternative quantum algorithms, such as QAOA, could be explored to solve the quadratic knapsack problem using the proposed QUBO formulations. Furthermore, the methods investigated in this study could be applied to other combinatorial problems, aiming to optimize their QUBO formulations and further advancing the field of quantum optimization.

ACKNOWLEDGEMENTS

This work was supported by grants TSI-064100-2023-34 funded by MINECO/SETeleco and EU-NextGenerationEU/PRTR; PID2022-140560OB-I00 (DRONAC) funded by MICIU/AEI /10.13039/501100011033 and ERDF, EU; MADQuantum-CM, financed by EU NextGeneration-EU, CM and PRTR; and grant CEX2024-001471-M funded by MICIU/AEI/10.13039/501100011033.

REFERENCES

- Awasthi, A., Bär, F., Doetsch, J., Ehm, H., Erdmann, M., Hess, M., Klepsch, J., Limacher, P. A., Luckow, A., Niedermeier, C., et al. (2023). Quantum computing techniques for multi-knapsack problems. In *Science and Information Conference* (pp. 264–284).: Springer.

- Ayodele, M. (2022). Comparing the digital annealer with classical evolutionary algorithm. *arXiv preprint arXiv:2205.13586*.
- Bontekoe, T., Phillipson, F., & Schoot, W. v. d. (2023). Translating constraints into qubos for the quadratic knapsack problem. In *International Conference on Computational Science* (pp. 90–107).: Springer.
- Boothby, T., King, A. D., & Roy, A. (2016). Fast clique minor generation in chimera qubit connectivity graphs. *Quantum Information Processing*, 15, 495–508.
- Borrajó, N., Ramírez, J. M., Nosrati, F., Aguilar, J., Mancuso, V., & Fernández Anta, A. (2025). New qubo transformations to improve quantum and simulated annealing performance for quadratic knapsack. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '25 Companion (pp. 203–206). New York, NY, USA: Association for Computing Machinery.
- Cai, J., Macready, W. G., & Roy, A. (2014). A practical heuristic for finding graph minors.
- Caprara, A., Pisinger, D., & Toth, P. (1999). Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2), 125–137.
- Coffey, M. W. (2017). Adiabatic quantum computing solution of the knapsack problem. *arXiv preprint arXiv:1701.05584*.
- D-Wave (2025a). D-wave ocean software documentation. Accessed: 2025-01-22.
- D-Wave (2025b). D-wave ocean software documentation - binaryquadraticmodel.add-linear-inequality-constraint. Accessed: 2025-01-22.
- Das, A. & Chakrabarti, B. K. (2008). Colloquium: Quantum annealing and analog quantum computation. *Reviews of Modern Physics*, 80(3), 1061–1081.
- Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm.
- Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., & Preda, D. (2001). A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516), 472–475.
- Finnila, A. B., Gomez, M. A., Sebenik, C., Stenson, C., & Doll, J. D. (1994). Quantum annealing: A new method for minimizing multidimensional functions. *Chemical physics letters*, 219(5-6), 343–348.
- Gallo, G., Hammer, P. L., & Simeone, B. (1980). Quadratic knapsack problems. *Combinatorial optimization*, (pp. 132–149).
- Glover, F., Kochenberger, G., Alidaee, B., & Amini, M. (2002). Solving quadratic knapsack problems by reformulation and tabu search: Single constraint case. In *Combinatorial and global optimization* (pp. 111–121). World Scientific.
- Glover, F., Kochenberger, G., Hennig, R., & Du, Y. (2022). Quantum bridge analytics i: a tutorial on formulating and using qubo models. *Annals of Operations Research*, 314(1), 141–183.
- Lucas, A. (2014). Ising formulations of many NP problems. *Frontiers in Physics*, 2.
- McGeoch, C. & Farré, P. (2022). Advantage processor overview. Accessed: 2025-01-22.
- Pisinger, D. & Toth, P. (1998). Knapsack problems. In D.-Z. Du & P. M. Pardalos (Eds.), *Handbook of Combinatorial Optimization: Volume 1–3* (pp. 299–428). Boston, MA: Springer.
- Pusey-Nazzaro, L. et al. (2020). Adiabatic quantum optimization fails to solve the knapsack problem. *arXiv preprint arXiv:2008.07456*.
- Quintero, R. A. & Zuluaga, L. F. (2021). Characterizing and benchmarking qubo reformulations of the knapsack problem. *Techn. Ber. Technical Report. Department of Industrial und Systems Engineering, Lehigh University*.
- Rajak, A., Suzuki, S., Dutta, A., & Chakrabarti, B. K. (2023). Quantum annealing: An overview. *Philosophical Transactions of the Royal Society A*, 381(2241), 20210417.
- Sao, M., Watanabe, H., Musha, Y., & Utsunomiya, A. (2019). Application of digital annealer for faster combinatorial optimization. *Fujitsu Scientific and Technical Journal*, 55(2), 45–51.
- Van Dam, W., Eldefrawy, K., Genise, N., & Parham, N. (2021). Quantum optimization heuristics with an application to knapsack problems. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* (pp. 160–170).: IEEE.
- van Laarhoven, P. J. M. & Aarts, E. H. L. (1987). *Simulated Annealing: Theory and Applications*, volume 37 of *Mathematics and Its Applications*. Springer.
- Verma, A. & Lewis, M. (2022). Penalty and partitioning techniques to improve performance of qubo solvers. *Discrete Optimization*, 44, 100594.