scientific reports

Check for updates

OPEN

QAL-BP: an augmented Lagrangian quantum approach for bin packing

Lorenzo Cellini[1✉], Antonio Macaluso[2] & Michele Lombardi[1]

The bin packing is a well-known NP-Hard problem in the domain of artificial intelligence, posing significant challenges in finding efficient solutions. Conversely, recent advancements in quantum technologies have shown promising potential for achieving substantial computational speedup, particularly in certain problem classes, such as combinatorial optimization. In this study, we introduce QAL-BP, a novel Quadratic Unconstrained Binary Optimization (QUBO) formulation designed specifically for bin packing and suitable for quantum computation. QAL-BP utilizes the Augmented Lagrangian method to incorporate the bin packing constraints into the objective function while also facilitating an analytical estimation of heuristic, but empirically robust, penalty multipliers. This approach leads to a more versatile and generalizable model that eliminates the need for empirically calculating instance-dependent Lagrangian coefficients, a requirement commonly encountered in alternative QUBO formulations for similar problems. To assess the effectiveness of our proposed approach, we conduct experiments on a set of bin packing instances using a real Quantum Annealing device. Additionally, we compare the results with those obtained from two different classical solvers, namely simulated annealing and Gurobi. The experimental findings not only confirm the correctness of the proposed formulation, but also demonstrate the potential of quantum computation in effectively solving the bin packing problem, particularly as more reliable quantum technology becomes available.

Bin packing is a well-established[1] combinatorial optimization problem with wide-ranging applications in domains such as logistics, resources allocation, and scheduling. Its primary objective is to minimize the number of fixed-capacity bins required to pack a set of items of variable size.

Despite extensive research efforts, bin packing remains a challenging problem due to the exponential growth of solution possibilities as the number of items and bins increases[2]. On the other hand, quantum computing has recently emerged as a promising alternative to solving various AI problems, including coalition formation in multi-agent systems[3–5] and supervised learning[6,7], although a real practical quantum advantage has yet to be found considering near-term quantum technology. The standard approach in quantum computing for optimization involves reformulating the original problem as a Quadratic Unconstrained Binary Optimization (QUBO) problem and employing quantum annealers (QAs) or parametrized quantum circuits, such as QAOA[8], to find the optimal solution. These approaches possess distinctive strengths and weaknesses. QAOA, for example, enables theoretical solutions to any QUBO problem with arbitrary precision by increasing the depth of the associated quantum circuit. On the other hand, QAs are specifically designed to identify the lower energy state of an Ising Hamiltonian representing the original QUBO problem and are better suited for tackling larger problems in terms of the number of QUBO variables.

In case of constraint optimization problems, the main drawback of the reformulation as a QUBO consists of associating a penalty term to the constraints and including them in the objective function. This approach requires empirical estimation of the penalty terms, which translates to running the QUBO solver (QA or QAOA) multiple times before achieving a feasible solution and poses several limitations to its real world applicability, especially when considering large problem instances and the imperfections of near-term quantum technology.

This paper presents a novel method for solving the bin packing problem using quantum computation. Specifically, we introduce an analytical heuristic approach for estimating penalty multipliers based on the Augmented Lagrangian framework[9], which allows to obtain a complete QUBO formulation without requiring empirical, instance-based parametrization. To demonstrate the effectiveness of our proposed method, we conduct experiments using a real quantum annealer and compare the results with two different state-of-the-art classical baselines.

[1]Department of Computer Science and Engineering, University of Bologna, Bologna, Italy. [2]Agents and Simulated Reality Department, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany. ✉email: lorenzo.cellini3@studio.unibo.it

nature portfolio

1

Problem formulation

The bin packing problem (BPP) is a classic optimization task that involves packing objects of different sizes into containers, or bins, with a limited capacity. The goal is to minimize the number of bins needed to pack all the objects. A mathematical formulation of the bin packing problem can be expressed as follows: given a set of n items of given integer size (or weight) w_j ($j = 1, \ldots, n$) the goal is to pack them into the minimum number of identical bins of integer capacity C.

Let m be any upper bound on the solution value and let introduce y_i, x_{ij} two sets of binary variables such that: $y_i (i = 1, \ldots, m)$ takes the value 1 if and only if bin i is used in the solution and $x_{ij} (i = 1, \ldots, m; j = 1, \ldots, n)$ takes the value 1 if and only if item j is packed into bin i. A commonly adopted Integer Linear Programming (ILP) problem formulation is the following[10]:

$$\arg \min_{x,y} \quad \sum_{i=1}^{m} y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = 1 \quad \forall j = 1, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} w_j x_{ij} \le C y_i \quad \forall i = 1, \ldots, m$$

$$x_{ij} \in \{0, 1\} \quad \forall i = 1, \ldots, m, \ \forall j = 1, \ldots, n$$

$$y_i \in \{0, 1\} \quad \forall i = 1, \ldots, m \tag{3}$$

A practical variant of significant interest is the *online* bin packing problem. In this scenario, items of varying sizes are observed sequentially, and the decision maker must determine whether to select and pack the currently observed item or let it pass. Each decision is made without the ability to recall previous decisions. In contrast, the *offline* bin packing problem allows for rearranging the items in an attempt to achieve a better packing arrangement when additional items arrive. However, this approach necessitates additional storage to hold the items that need to be rearranged.

Related works

Classical algorithms for solving the BPP rely on Linear Programming relaxations and dynamic programming[11,12]. However, as the number of items increases, the problem becomes intractable, and even for medium-sized instances, the optimal solution cannot be computed within a reasonable time frame. For this reason, several approximation algorithms and heuristics approaches can be adopted, such as simulated annealing[13,14], Tabu search[15], population-based algorithms[16], evolutionary and genetic heuristics[17–21] with hyper-heuristics[22–25], variable neighborhood search meta-heuristics[26] and ad-hoc crafted heuristics[27–29].

In addition various alternative reformulations of the BPP have been proposed to improve the computational performance, such as pseudo-polynomial models[30–32]. While these approaches offer more efficient problem formulations and enable the implementation of solutions that do not scale exponentially with the input size, they suffer from the drawback of the number of variables depending on both the number of items and the bin capacity.

More recently, the adoption of quantum computing has been explored for solving BPP. Existing quantum solutions involve reformulating the original problem as a QUBO problem and leveraging gate-based quantum computers or quantum annealing.

At the time of writing this paper, two end-to-end QUBO models have been proposed for BPP, namely the Pseudo-Polynomial formulation[33] and the Unbalanced Penalization approach[34]. Alternatively, another existing approach[35] addresses the BPP through a hybrid approach, using quantum annealing to solve the sub-problem of filling a single bin, with the chance of reaching a sub-optimal solution.

Pseudo-polynomial QUBO formulation

The Pseudo-Polynomial QUBO formulation for the BPP[33] is defined by three sets of binary variables. These variables are employed to represent the placement of weights in bins, indicate whether bins are empty or not, and specify the filling levels of the bins. The corresponding Hamiltonian of the QUBO formulation consists of two weighted components, necessitating empirical estimation for the weights for each problem instance to avoid infeasible solutions. The primary objective of the Hamiltonian is to minimize the number of used bins, which constitutes the classical objective function. The constraints Hamiltonian comprises three components. Firstly, it enforces the condition that each bin must be filled up to a unique level while ensuring that unused bins remain unfilled. Secondly, it guarantees that every item is allocated to a bin. Lastly, the third component penalizes configurations in which bins are overfilled, thereby violating the capacity constraint. Additionally, an extra term is introduced to account for considerations regarding only non-empty bins.

A significant drawback of this formulation is its limited scalability concerning the number of binary variables. Although the introduction of slack variables enables the formulation to be pseudo-polynomial, it results in the addition of nC binary variables, making the formulation dependent on the specific problem instance due to the influence of bin capacity on the variable count. This scalability issue becomes particularly pertinent in the context of modern Quantum Processing Units (QPUs), which face restrictions in handling relatively small problem

instances due to qubit topology and connectivity constraints[36]. Consequently, even for small problem instances, a pseudo-polynomial Hamiltonian may become intractable when implemented on such quantum platforms.

Moreover, achieving a well-balanced weight assignment for each term in the Hamiltonian is of paramount importance to effectively minimize the number of used bins while satisfying the defined constraints. This requires to run the same problem instance multiple times using various hyperparameter sets in order to optimize the formulation's performance.

Unbalanced penalization formulation
An alternative QUBO model for the BPP[34] introduces an inequality constraint $g(x) = \sum_i l_i x_i - C \leq 0$ whose violation can be penalized using the exponential function $e^{g(x)}$. To ensure a valid QUBO model, the exponential function is expanded up to its second-order Taylor's term, resulting in the approximation $e^{g(x)} \approx 1 + g(x) + \frac{1}{2}g(x)^2$.

Despite its improved efficiency compared to the pseudo-polynomial QUBO formulation, this work has several limitations. Firstly, this formulation requires the estimation of lambda parameters (that act as constraints multipliers) specific for each problem instance. This implies running the quantum algorithm multiple times to obtain feasible solutions for a single problem instance.

Secondly, the model's performance is evaluated on a limited set of problem instances, raising concerns about its generalizability to other instances. The model's scalability across instances with varying numbers of items is not demonstrated; the evaluation is restricted to randomly generated instances with the same number of items.

Furthermore, the experimental testing of the solution relies on QAOA, which poses restrictions on the number of problem variables due to the challenges associated with simulating even small-scale quantum systems.

Contribution
In this work, we present QAL-BP (*Quantum Augmented Lagrangian method for Bin Packing*), a novel QUBO formulation for the BPP based on the Augmented Lagrangian method. QAL-BP is an end-to-end method for deriving a QUBO formulation for the BPP that enables efficient scaling of logical qubits and the analytical estimation of the Lagrangian penalty terms. Specifically, we establish a connection between QUBO models and Augmented Lagrangian methods, leveraging advancements in both fields and fostering potential future synergies.

The proposed formulation offers several advantages. Firstly, it exhibits independence of the number of variables from the bin capacity. This eliminates the need for introducing slack variables, which typically increase the number of logical qubits and make quantum solutions infeasible for execution on real quantum hardware. Secondly, we analytically determine the Lagrangian penalty terms for a specific class of instances, avoiding running the quantum annealing multiple times on the same instance, as it is usually necessary for alternative approaches. Through experiments conducted on a real quantum annealing device, we demonstrate the effectiveness of our proposed approach. Thirdly, we compare the performance of QAL-BP with state-of-the-art classical approaches. The results demonstrate that QAL-BP consistently yields feasible solutions, and in most cases, it leads to the global minimum.

To the best of our knowledge, this marks the first instance of an end-to-end analytical quantum solution for the BPP that has been rigorously tested across a diverse set of problem instances, displaying superior performance in comparison to existing quantum solutions. Furthermore, our results indicate promising potential concerning state-of-the-art classical solvers, particularly when more reliable quantum devices will become available.

Methods
QAL-BP: quantum augmented Lagrangian method for bin packing
Augmented Lagrangian methods are a class of algorithms used to solve constrained optimization problems by incorporating the constraints into the objective function through penalty terms. Consider a constrained minimization problem of the form:

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t.} \quad c_i(\mathbf{x}) = \mathbf{b} \; \forall i \in \mathcal{D} \end{aligned} \tag{4}$$

where x is a candidate solution, $c_i(\mathbf{x}) = \mathbf{b}$ are a set of equality constraints and $\mathcal{D}$ is the set of constraints. The Augmented Lagrangian method consists in defining an unconstrained problem of the form:

$$\min \Phi(\mathbf{x}) = f(\mathbf{x}) + \sum_{i \in \mathcal{D}} \rho_i (c_i(\mathbf{x}) - \mathbf{b})^2 + \sum_{i \in \mathcal{D}} \lambda_i (c_i(\mathbf{x}) - \mathbf{b}) \tag{5}$$

where ρ_i, λ_i for $\{i = 1, ..., |\mathcal{D}|\}$ are the lagrangian multipliers.

In practice, when using the Augmented Lagrangian approach, it is common to introduce additional constraints that do not alter the set of feasible solutions but aid in faster convergence of the solver[37].

For the BPP, we introduce the following additional constraints:

$$\sum_{j=1}^{n} y_i x_{ij} = \sum_{j=1}^{n} x_{ij} \quad \forall i = 1, \ldots, m \tag{6}$$

This set of constraints imply that if a bin i is not used ($y_i = 0$), it cannot contain any items ($\sum_{j=1}^{n} x_{ij} = 0$).

Therefore, the Quantum Augmented Lagrangian method for Bin Packing (QAL-BP) embeds the constraints using the Augmented Lagrangian approach as follows:

$$\arg \min_{x,y} \delta \sum_{i=1}^{m} y_i \tag{7}$$

$$+ \sum_{i=1}^{m} \lambda_i \left(\sum_{j=1}^{n} w_j x_{ij} - c_i y_i \right) \tag{8}$$

$$+ \sum_{i=1}^{m} \rho_i \left(\sum_{j=1}^{n} w_j x_{ij} - c_i y_i \right)^2 \tag{9}$$

$$+ \theta \sum_{j=1}^{n} \left(\sum_{i=1}^{m} x_{ij} - 1 \right)^2 \tag{10}$$

$$+ \gamma \sum_{i=1}^{m} \left(1 - y_i \right) \sum_{j=1}^{n} x_{ij} \tag{11}$$

$$\text{s.t.} x_{ij} \in \{0, 1\} \quad \forall i = 1 \ldots, m, \forall j = 1 \ldots, n$$
$$y_i \in \{0, 1\} \quad \forall i = 1 \ldots, m \tag{12}$$
$$\delta, \quad \lambda, \rho, \theta, \gamma \geq 0$$

The penalties (8) and (9) represent the Augmented Lagrangian expansion of (3). These terms impose a penalty of $\lambda_i s_i + \rho_i s_i^2$ for infeasible configurations while providing a negative reward to the solver for feasible configurations. Accurate estimation of λ_i and ρ_i values is crucial for correctly modeling the solution space.

Similarly, in Eq. (10), θ represents a penalty for not placing an item j, and penalizes $(k-1)\theta$ when item j is placed k times. Notably, this penalty term is not an Augmented Lagrangian expansion of (2), but rather a pure squared penalty. This is because we do not wish to reward the solver when an item is not placed at all.

Finally, the term (11) represents the penalty associated with the redundant constraints (6) which imposes a penalty of $\sum_{j \in \mathcal{J}} x_{ij}$ when the set of items $\mathcal{J}$ is assigned to bin i without setting the corresponding y_i to 1.

It is important to note that the standard Augmented Lagrangian approach typically transforms inequality constraints into equality constraints using slack variables, which are then incorporated into the Lagrangian as shown in (5). However, the proposed QUBO formulation in this study does not involve slack variables but directly utilizes the capacity constants c_i. This aspect provides a significant advantage over the pseudo-polynomial approach[33].

Penalties estimation

When incorporating constraints into the objective function, the estimation of penalty multipliers is typically carried out by testing a large set of parameters which requires running the algorithm multiple times with different parameter values in order to find an optimal solution for the specific problem instance, thus dramatically increasing the cost (in terms of time) to find good solutions.

Given the QAL-BP formulation and its corresponding set of constraints, we propose an analytical estimation of the penalty multipliers. The conditions are designed based on approximate worst-case reasoning, aiming to yield optimal or slightly sub-optimal solutions for most instances of the BPP. The following is a set of heuristic conditions that consider each penalty multiplier individually, with the exception of the pair λ, ρ. Considering the i-th bin, the correspondent Augmented Lagrangian term is given by $\lambda_i(s_i - c_i y_i) + \rho_i(s_i - c_i y_i)^2$. When $y_i = 0$, i.e, when the bin i is not included in the final solution, using the smallest bin usage amount should be at least as expensive as using the bin:

$$\lambda_i(w_{min} - 0) + \rho_i(w_{min} - 0)^2 \geq 1, \tag{13}$$

where w_{min} is the smallest item weight, i.e. $w_{min} = \min\{w_j\}$. If this condition is satisfied, using more capacity makes the solver choose to set $y_i = 1$. On the contrary, if $y_i = 0$ and no capacity is used, then the condition in eq. (13) is trivially true. Let's consider now the case where $y_i = 1$, meaning that the bin y_i is included in the candidate solution. In this case, exceeding the capacity by any amount should be at least as expensive as using one more bin, i.e.:

$$\lambda_i(c_i + w_{min} - c_i) + \rho_i(c_i + w_{min} - c_i)^2 \geq 1 \tag{14}$$

which conveniently is the same condition as Eq. (13). Lastly, it is necessary to identify a solution space that contains only feasible solutions. In this case, the Lagrangian term needs to provide a positive reward (i.e., negative cost) if the constraint is satisfied. It is also necessary for such a reward to be small enough that so it does not provide an incentive for using another bin, i.e.:

$$\lambda_i\left(-\frac{c_i}{2}\right) + \rho_i\left(-\frac{c_i}{2}\right)^2 \geq 0 \tag{15}$$

Basically, here we are fitting a quadratic function using the conditions (13) and (15) to approximate the solution space excluding infeasible solutions. We can therefore obtain values for λ_i and ρ_i by stating all conditions for their least restrictive values:

$$w_{min}\lambda_i + w_{min}^2\rho_i = 1; \tag{16}$$

$$-\frac{c_i}{2}\lambda_i + \frac{c_i^2}{4}\rho_i = 0, \tag{17}$$

which leads to an analytical formulation of the form:

$$\lambda_i = \frac{c_i}{w_{min}(2w_{min} + c_i)} \tag{18}$$

$$\rho_i = \lambda_i \frac{2}{c_i} = \frac{2}{w_{min}(2w_{min} + c_i)} \tag{19}$$

The next step is to calibrate θ_j. The abstract Lagrangian term associated with item j is $\theta_j(p_j - 1)^2$, where $p_j \in \mathbb{N}$ is the number of times item j has been assigned to a bin. At this stage, only the assignment is considered and not capacities, so it is possible to ignore the index j when defining the θ parameter. Moreover, we want to force the solver to assign all items to at maximum one bin, so the penalty should increase when $p_j \neq 1$. Thus, in case $p_j = 1$, the following condition holds:

$$\theta(1 - 1)^2 \geq -1. \tag{20}$$

In case $p_j = 0$, i.e., item j is not assigned to any bin, then the associated penalty needs to be greater than the cost of opening a new bin, i.e.,

$$\theta \geq 2. \tag{21}$$

Also, the parameter γ needs to be set. The abstract Lagrangian term associated with the γ term is $\gamma(1 - y_i)k_i$, which comes into play only when $y_i = 0$ and $k_i \neq 0$ by adding k_i times the penalty γ. We want the minimum penalty to be at least equal to the cost of opening a new bin, i.e.,

$$\gamma \geq 1. \tag{22}$$

The final parameter to be estimated is δ. Although it is not strictly a penalty term since it serves as a multiplier of the objective function, including it is beneficial for controlling the other parameters. One purpose of δ is to prevent issues that may arise when working with very small numbers.

Another reason for incorporating this multiplier is to address the undesirable behavior of the model in certain item configurations. In some cases, the model may favor configurations where one or more bins are slightly overfilled due to the high cost associated with opening a new bin. Naturally, this behavior is contingent on the specific combination of instance weights relative to the bin capacity, as well as their number.

To rectify this behavior, the following requirements must be met:

$$\delta \leq \lambda s_{min} + \rho s_{min}^2, \tag{23}$$

where s_{min} is the minimum capacity that can be exceeded. Thus, the cost of opening a new bin must be less than the cost of overfilling an already open bin, of the smallest possible amount, i.e., $s_{min} \geq 1$.

Model analysis

When solving QUBO problems with quantum computing, the number of binary variables of the problem formulation corresponds to the number of logical qubits to use in the quantum computer. Therefore, having an efficient formulation that minimizes the number of variables without limiting the range of possible solutions is crucial for the adoption of quantum approaches. In terms of variable count, the QAL-BP model is more efficient than the pseudo-polynomial one[33], and equivalent, with respect to the *Unbalanced penalization formulation*[34]. Specifically, for a given problem instance $BPP(n, C)$, the total number of variables is equal to the number of bins m plus $n \times m$ decision variables representing the assignment of a specific item to a specific bin. Thus, in the worst case, $m = n$, resulting in $n(n + 1)$ binary variables.

Furthermore, from a methodological point of view, the QAL-BP approach offers a twofold advantage. Firstly, the number of binary variables is not affected by bin capacities and item weights, as observed in the pseudo-polynomial formulation[38]. Secondly, the reduced number of variables enables the execution of the QUBO problem using a smaller set of logical qubits, making it suitable for current QPUs.

Fig. 1 shows a comparison of the performance, in terms of the number of variables, between the QAL-BP and the pseudo-polynomial formulation.
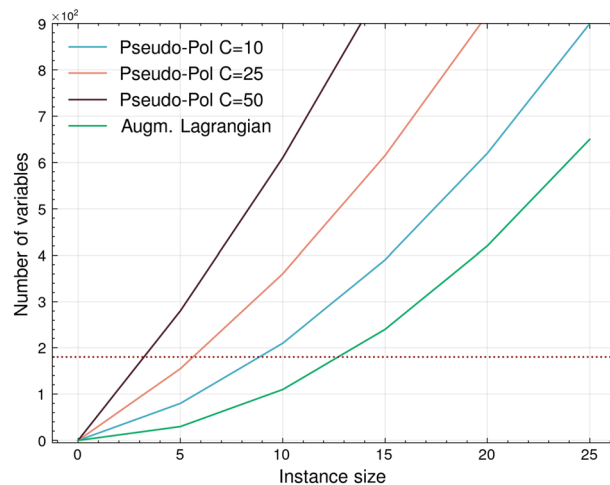
Figure 1. Comparative analysis of variable growth in the Pseudo-Polynomial and Augmented Lagrangian models concerning the number of items and bin capacity. Three distinct values of bin capacity (C) are explored. The continuous dark red line represents the upper limit for QUBO problems represented by fully connected graphs that can be mapped in the D-Wave Advantage Quantum Processing Unit (QPU) equipped with 5640 qubits.

Evaluation
Experimental settings
Data

The experiments are performed on a set of eight classes of randomly generated instances, ranging from 3 to 10 items, with corresponding weights ranging from 4 to 10 and fixed bin capacity equal to 10 (Table 1). This choice allows exploring problem instances of different sizes while taking into account the limitations of modern QPUs which are restricted to non-sparse QUBO problems with up to 180 binary variables[36]. Five different problem instances are generated for a fixed number of items, that vary in the weights.

Solving methods

The quantum solver employed in this study is the *D-Wave Advantage 4.1*, featuring a total of 5640 physical qubits. For comparison, two classical solvers are utilized. The first is *simulated annealing* (SA), which is considered the classical counterpart of quantum annealing, enabling a thorough assessment of the correctness of the QAL-BP formulation without encountering any errors inherent to real quantum hardware. Both quantum annealing and simulated annealing are available in the *D-Wave Python library*[40]. In addition, we also solve the ILP formulation from Eqs. (1)–(3) via *Gurobi* optimizer[41], as a representative classical state-of-the-art method that relies on the branch-and-bound technique to efficiently find the optimal solution.

As previously mentioned, the rationale behind choosing quantum annealing over alternative quantum approaches, such as QAOA, is to enable a direct performance comparison between the most powerful and dependable current quantum technology and the state-of-the-art classical optimizer. This comparison aims to demonstrate the capability of current quantum computation in solving the BPP in relation to the best available classical solution. It is important to recognize that quantum hardware is still in its nascent stage, making such a comparative analysis critical in assessing the advancements and potential of quantum computing in tackling optimization problems like BPP.

Metrics

In order to assess the performance of QAL-BP, we consider four distinct aspects: *scalability*, *runtime*, *solution quality*, and *feasibility ratio*.

Regarding *scalability*, our investigation encompasses a thorough examination of the necessary physical qubits for QAL-BP. In fact, when using a real quantum annealing device, a single logical variable (or logical qubit) in the QUBO problem is typically represented by a set of physical qubits within the real hardware, by means of chains. This is due to the incomplete connectivity of the QPU topology. The representation of a variable through a chain imposes the condition that all constituent qubits must take identical values for a given sample. Achieving this uniformity is contingent upon establishing robust couplings between the qubits along the connecting edges. Specifically, a strong coupling is implemented to ensure that the qubits in a chain are forced to return identical values and is controlled via a parameter called *chain_strength*. This metric is used to strengthen the coupling between qubits belonging to the same chain, against other topologically close qubits that may exert influence leading to divergent outcomes. In this context, we examine the `chain_break_fraction`, which serves to quantify the fraction of chains in a sample that experiences disruption. A small `chain_break_fraction` is indicative of a more favorable outcome. Additionally, we implement different strategies to mitigate the chain

Seed	N. items	Weights	Lower bound	Instance name
23	3	[4, 8, 6]	2	(3, 23)
	4	[8, 5, 4, 8]	3	(4, 23)
	5	[4, 4, 8, 8, 9]	3	(5, 23)
	6	[7, 5, 5, 5, 4, 9]	4	(6, 23)
	7	[9, 7, 8, 6, 9, 6, 7]	5	(7, 23)
	8	[4, 5, 7, 5, 6, 4, 6, 4]	4	(8, 23)
	9	[7, 6, 8, 4, 8, 4, 9, 6, 4]	6	(9, 23)
	10	[5, 8, 6, 7, 10, 9, 4, 10, 7, 4]	7	(10, 23)
42	3	[4, 8, 6]	2	(3, 42)
	4	[7, 7, 10, 4]	3	(4, 42)
	5	[8, 5, 4, 7, 10]	4	(5, 42)
	6	[9, 9, 9, 9, 7, 4]	5	(6, 42)
	7	[9, 7, 7, 6, 5, 10, 9]	5	(7,42)
	8	[8, 6, 9, 7, 7, 7, 5, 4]	5	(8, 42)
	9	[7, 10, 4, 10, 9, 5, 8, 5, 9]	7	(9, 42)
	10	[8, 6, 4, 10, 7, 10, 8, 9, 9, 5]	7	(10, 42)
123	3	[4, 8, 8]	2	(3, 123)
	4	[4, 10, 5, 5]	3	(4, 123)
	5	[5, 6, 5, 6, 9]	3	(5, 123)
	6	[7, 10, 7, 5, 9, 9]	5	(6, 123)
	7	[10, 10, 4, 7, 5, 5, 5]	5	(7, 123)
	8	[9, 9, 5, 6, 9, 5, 8, 7]	6	(8, 123)
	9	[10, 9, 5, 9, 9, 5, 7, 9, 5]	7	(9, 123)
	10	[5, 5, 4, 7, 4, 8, 6, 5, 6, 4]	5	(10, 123)
90	3	[8, 6, 4]	2	(3, 90)
	4	[8, 5, 7, 6]	3	(4, 90)
	5	[6, 7, 8, 7, 4]	3	(5, 90)
	6	[7, 8, 9, 9, 10, 6]	5	(6, 90)
	7	[6, 4, 4, 4, 8, 9, 6]	4	(7, 90)
	8	[7, 10, 8, 8, 8, 5, 5, 8]	6	(8, 90)
	9	[9, 6, 4, 10, 10, 5, 4, 4, 6]	6	(9, 90)
	10	[9, 6, 8, 7, 8, 10, 9, 6, 9, 10]	8	(10, 90)
510	3	[5, 8, 6]	2	(3, 510)
	4	[7, 9, 5, 5]	3	(4, 510)
	5	[6, 10, 4, 9, 4]	3	(5, 510)
	6	[5, 5, 9, 10, 8, 6]	4	(6, 510)
	7	[9, 7, 9, 4, 10, 10, 8]	6	(7, 510)
	8	[9, 10, 8, 9, 4, 4, 9, 5]	6	(8, 510)
	9	[5, 9, 10, 9, 7, 8, 4, 10, 6]	7	(9, 510)
	10	[10, 5, 9, 5, 8, 9, 7, 4, 6, 9]	7	(10, 510)

Table 1. Bin Packing problem instances. Each instance is characterized by specific parameters, which are organized into columns. The first column denotes the numeric seed utilized for generating the instances. The second column indicates the number of items that need to be placed within the bins. The third column comprises an array representing the weight of each individual item. The fourth column provides the L_1 lower bound for the given instance[39]. The last column assigns a unique label to facilitate identification and reference to each specific instance.

break effect. Each strategy corresponds to different values of *chain_strength*. The first strategy, *uniform_torque_compensation*, aims to compensate for the random torque of neighboring qubits that could potentially break the chain. The prefactor is set to the default value of 1.414, resulting in a chain strength in the range 2.3–5.5. The second strategy, *scaled*, involves adjusting the chain strength to align with the problem bias, which is defined by the values taken by the linear and quadratic terms in the corresponding Ising problem. This calibration ensures a chain strength between 2.16 and 16.5. The third strategy, *constant_10*, selects a constant value of 10 based on reasoning provided in a D-Wave whitepaper[42]. The fourth strategy, *constant_6*, employs a constant value of 6, chosen by rounding up the maximum chain strength found by the *uniform_torque_compensation* technique.

The *runtime* is described in terms of Time-To-Solution (TTS), denoting the duration required by the solver to generate the ultimate solution, with measurements expressed in microseconds (μs). For QA, we sum up two

different metrics: the *qpu_sampling_time* and the time for *embedding*. The variable *qpu_sampling_time* serves as an indicator of the active problem-solving duration of the QPU. Also, we incorporate the *embedding* time, signifying the process of mapping an arbitrarily posed binary quadratic problem directly onto the QPU topology of an actual D-Wave system. Both *qpu_sampling_time* and *embedding* stand as the two primary components requiring consideration in evaluating QA, as their impact is contingent upon the specific hardware in use and the nature of the QUBO problem to be solved. For comprehensive coverage, we also present an in-depth description of all the metrics supplied by D-Wave, emphasizing that their influence is negligible when evaluating QA computations. Differently from QA, SA and Gurobi run locally. Thus, the TTS of SA is the time to run the annealing function, while for Gurobi, TTS is calculated as the time required to obtain the solution for a given problem instance.

The *quality* of a solution is defined by the number of bins utilized in a given configuration, with a lower count of bins indicating a superior solution.

The *feasibility ratio* serves as a quantitative measure of the probability that the solution with the minimum energy aligns with a feasible outcome. More precisely, this ratio is computed as the quotient between the number of instances in which the solution with minimum energy corresponds to a feasible solution (not necessarily the global optimum) and the total number of instances within each class of instances. This metric holds significance exclusively within the context of simulated and quantum annealing, enabling an evaluation of the efficacy of contemporary quantum technology in providing solutions for the QAL-BP formulation.

Models parameters
According to the analytical penalty estimation previously described, the multipliers are determined as follows: $\delta = 0.15$; $\lambda = 0.1389$; $\rho = 0.0278$; $\theta = 2$; $\gamma = 1$.

Results
Scalability
Figure 2 illustrates the required number of physical qubits for embedding BPP instances on a QPU. It is evident that the QAL-BP formulation necessitates an exponential increase in the number of physical qubits in comparison to the number of logical qubits (i.e., the number of variables in the QUBO). This characteristic represents a notable limitation when attempting to apply the formulation to real-world problems using existing quantum hardware.

Additionally, the inherent limited connectivity of the QPU's topology introduces a vulnerability through coupling chains that connect the physical qubits encoding a single logical variable. Figure 3 offers insight into the distribution of the `chain_break_fraction` for each instance and its consequential impact on the solutions derived from QA.

For problem instances involving 3 to 7 items, the `chain_break_fraction` is null, and the QA yields identical solutions to SA. This implies that QA successfully identifies the global optimum within the QAL-BP formulation. Nevertheless, when the number of items equals or exceeds 8, the energy associated with SA solutions consistently surpasses that of QA. This persists even when the `chain_break_fraction` is null for four out of five instances with 8 items, although the observed difference is relatively small in this specific case. Upon closer examination of chain breaks in instances involving more than 9 items, the QA solutions markedly deteriorate compared to those produced by the SA. Notably, in all these cases, the `chain_break_fraction` consistently exceeds 0. Hence, while the occurrence of chain breaks undeniably impacts solution quality, it is not the sole determinant preventing QA from returning to the global optimum, as evidenced by instances involving 8 items.

The effect of chain breaks can be alleviated through the adjustment of *chain_strength* during the embedding of the QUBO problem into the topology of a real quantum annealer. To address this, we conduct a set of experiments testing four distinct strategies to determine the optimal *chain_strength*[43]. Results are shown in Figure 4.

We observe a correlation between the increase in chain strength and a corresponding increase in energy, potentially leading to a reduction in the number of successfully solved problems. This phenomenon is explained
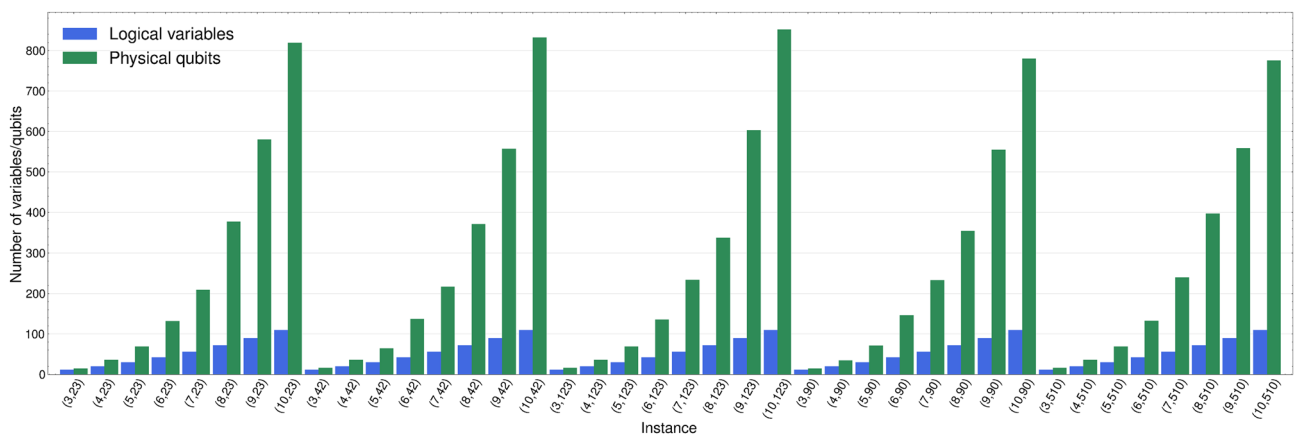

Figure 2. Comparison of the number of logical variables of QAL-BP against the physical qubits needed to implement the QUBO problem on the *D-Wave Advantage 4.1*.
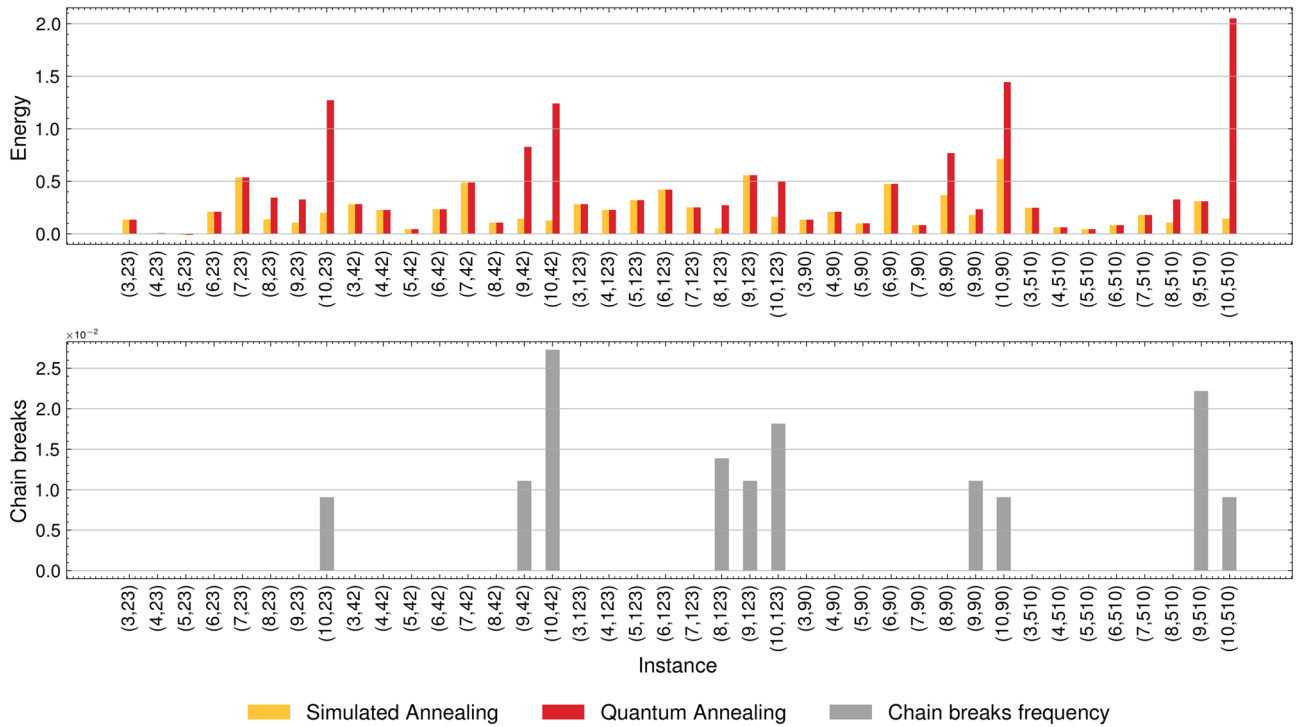
Figure 3. Comparative evaluation of the energy of the solutions provided by SA and QA (top), and the `chain_break_fraction` reported by the QA implementation (bottom).
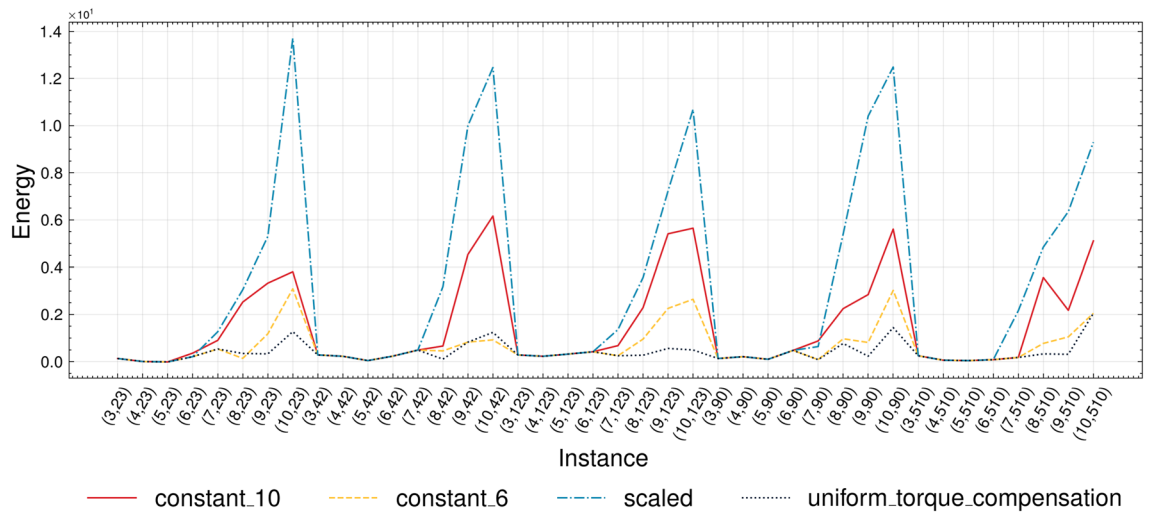

Figure 4. Comparative evaluation of the energy values for different chain break mitigation strategies.

by the fact that when submitting a problem to the QPU, the auto-scaling feature scales all QUBO weights in the range $[-1, +1]$. If the chain strength is too large, it causes individual QUBO coefficients, designed to control both the terms requiring minimization and the problem constraints, to shrink toward near-zero values. Consequently, each chain assumes an independent status, transforming the QUBO into a problem characterized by independent variables that lack mutual interactions[42].

To identify the most effective mitigation strategies, we solved all instances previously described and calculated, for each, the percentage of cases where the optimal solution with minimum energy aligns with a feasible solution. The results are as follows: 52.5% for *scaled*, 57.5% for *constant_10*, 67.5% for *constant_6*, and 82.5% for *uniform_torque_compensation*. Based on these results, we designate *uniform_torque_compensation* as the most effective mitigation strategy for QAL-BP. Consequently, this strategy is employed in all subsequent experiments discussed throughout the remainder of this paper.

Runtime

Figure 5 illustrates the experimental results concerning TTS. The performance of SA deteriorates rapidly with the number of items, becoming extremely inefficient even for relatively small problems.

In the comparative analysis between QA and Gurobi, the latter consistently exhibits superior performance, although the asymptotic behavior seems to be equivalent between the two methods. However, this might be due to the relatively small problem instances that the classical optimizer can easily solve. In fact, the Gurobi relies on an exact branch-and-bound approach[44], which in the worst case ends up exploring the entire solution space, resulting in prohibitive runtime when dealing with large problems. Nevertheless, the QA has demonstrated the capacity to scale more efficiently than Gurobi for very large problems[45], with a runtime linear in the input size[3,4]. Taking these considerations into account, the findings suggest the potential for QA to surpass Gurobi's performance as more reliable quantum technology becomes available. Nonetheless, due to constraints in running larger problem instances, a definitive assessment of performance remains elusive.

Furthermore, we present the averages across all problem instances for various metrics offered by the D-Wave platform[40]. As depicted in Figure 6, the TTS associated with annealing primarily relies on the *sampling_time* that
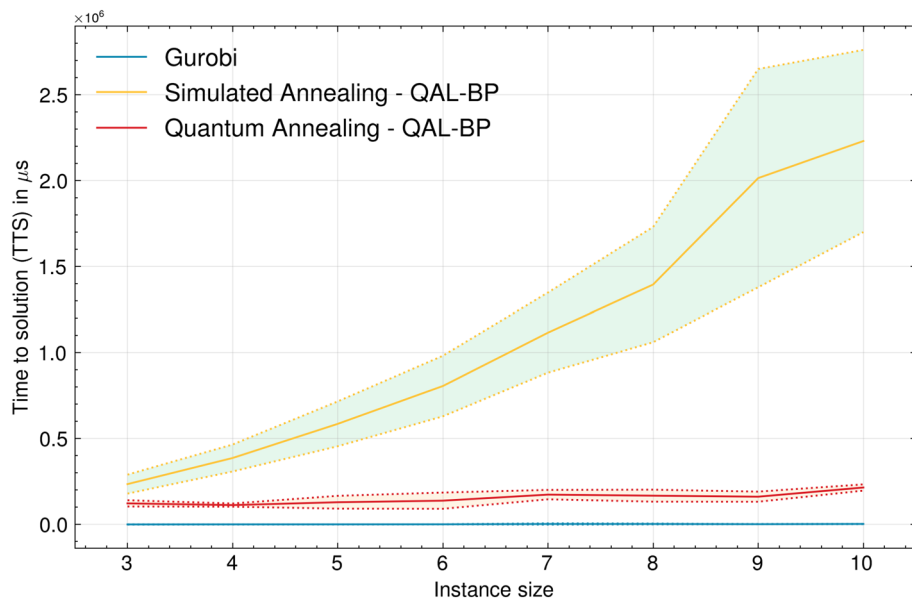

Figure 5. TTS comparison between Quantum Annealing (QA), Simulated Annealing (SA) and Gurobi. The Gurobi curve ranges between $440\mu s$ and $3000\mu s$.
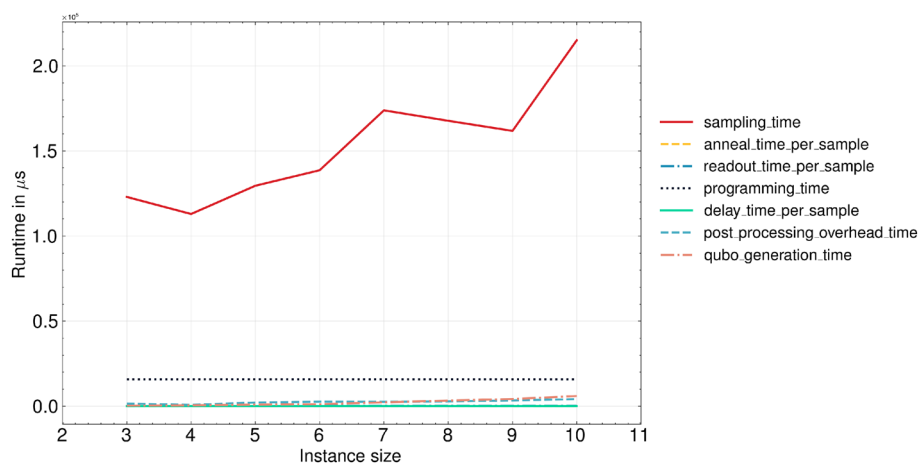

Figure 6. Runtime breakdown of the D-Wave QPU when solving the QUBO problem according to the QAL-BP formulation. Each metric is the average over all the instances of the same class, i.e., problems with the same number of input bins. In addition to the metrics provided by the D-Wave Python Library, we include the average time per instance class necessary to transform the ILP formulation into QAL-BP (*qubo_generation_time*).

encompasses the anneal-read cycle, iterated for a specified number of samples, as determined by the *num_reads* parameter (in our case this number is set to 1000).

Solution quality

In terms of *solution quality*, i.e., the number of bins used by a specific solution, Figure 7 shows the results provided by the three solvers adopted.

Gurobi solutions act as a benchmark, representing a well-established deterministic optimizer that correctly outputs the minimum required number of bins while explicitly taking into consideration all the constraints. The outcomes derived from SA validate the reliability of the QAL-BP formulation in consistently identifying feasible solutions for various problem instances. In the majority of cases, the optimal solution produced by SA aligns with the global optimum of the Gurobi. The only exceptions are noted in instances (8, 23), (10, 123), and (9, 510), where the solutions generated by SA, while feasible, are characterized by a higher utilization of bins.

Concerning the outcomes derived from QA, it is evident that, in the majority of cases, the solutions align with those provided by SA. Deviations manifest in three potential scenarios. Firstly, the QA-generated solutions are suboptimal compared to SA, implying the utilization of a higher number of bins, as observed in (8, 23). In such instances, the constraints imposed by the application of real quantum hardware lead to a feasible solution that, while not reaching the global optimum, remains within the bounds of feasibility. Secondly, instances of infeasible solutions may arise, exemplified by cases like (10, 23), (9, 42), (10, 90), and (10, 510), wherein the selected number of bins falls below the Gurobi solution, signaling the occurrence of overfilled bins. Indeed, because Gurobi explicitly considers the constraint on bin capacity, the number of bins yielded by it stands as the global optima for the given problems. Therefore, the lower count of bins in the QA solution indicates inadequacy, signifying an overfilled state for the utilized bins, as they fail to contain the items adequately. Thirdly, instances may arise where the selected number of bins appears to be appropriate based on the count of bins alone but proves to be infeasible due to violations of other constraints, e.g. instances (10, 42) and (8, 90) where, although the correct number of bins is selected, some of them have been overfilled. It is worth to notice that SA, employing the QAL-BP formulation, produces feasible solutions for the same problem instances. This substantiates that the infeasible solutions generated by QA are attributable to hardware errors rather than inherent deficiencies in the proposed formulation.

Feasibility ratio

To further highlight the challenges associated with the current QA hardware in accurately determining solutions for the QAL-BP formulation, Figure 8 depicts the *feasibility ratio* across various instances of identical size. In the context of SA, solutions consistently manifest feasibility, as indicated by a *feasibility ratio* of 1. This observation underscores that the QAL-BP formulation provides correct solutions for the BPP.

Concerning the QA implementation, for instances comprising up to 7 items, QA uniformly yields feasible solutions in all cases. However, for instances involving 8 and 9 items, the *feasibility ratio* diminishes to 80%. This reduction indicates that, in one out of the five instances tested with the same item count, the solution with minimum energy is infeasible. In the 100% of instances of 10 items, instead, QA was unable to find a feasible solution. Importantly, the observed decline in the *feasibility ratio* appears to correspond to an increase in the `chain_break_fraction`.

In summary, the experimental findings illustrate that the QAL-BP formulation facilitates precise analytical estimation of penalty parameters, yielding near-global optimum solutions in all instances. However, the adoption of a real quantum annealer for the QUBO formulation of the BPP reveals suboptimal performances for medium-sized problems, attributable to existing limitations in current quantum hardware. Consequently, to fully harness the potential of the proposed QAL-BP method in effectively addressing larger instances of the BPP, imperative advancements in quantum technology and solver optimization are required.
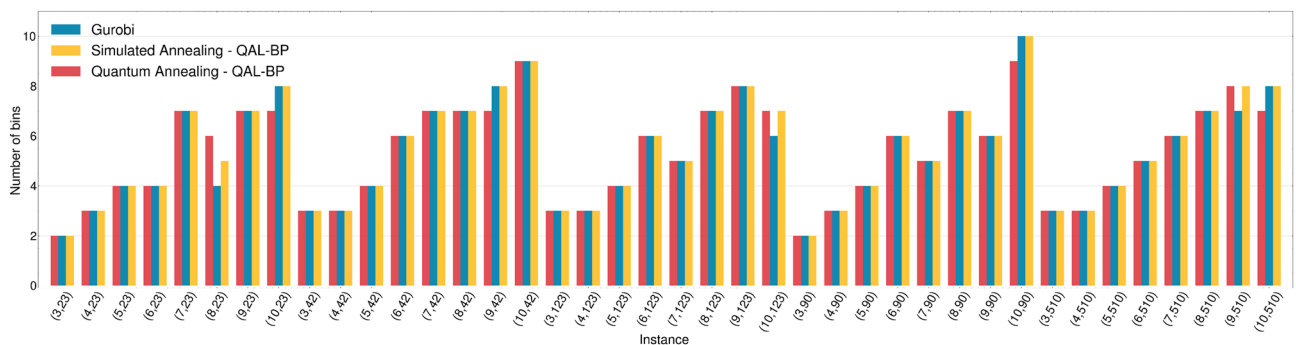

Figure 7. Comparison of the number of bins corresponding to the solution with the minimum energy, for each problem instance, found by Gurobi, Simulated and Quantum annealing solvers.
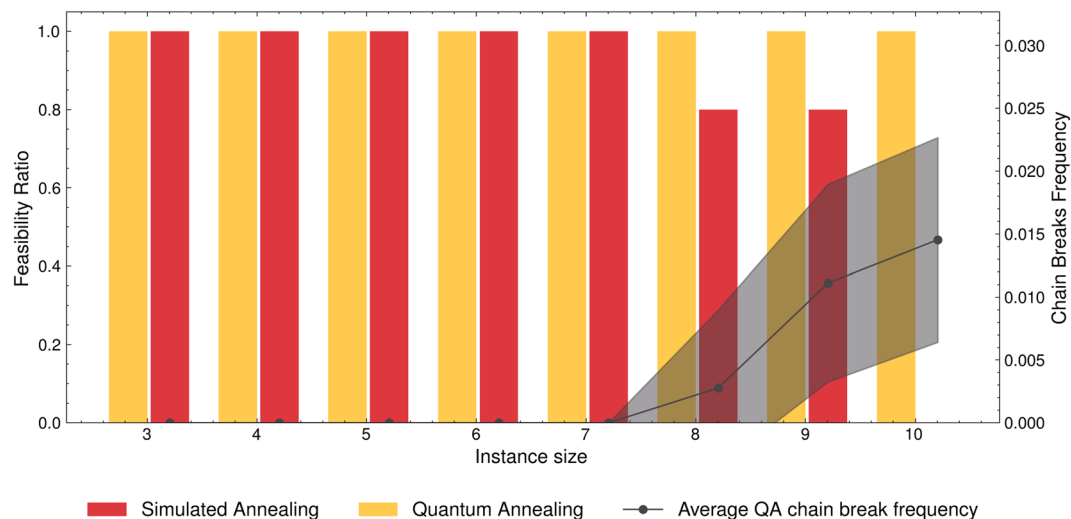
Figure 8. Comparison of the *feasibility ratio* for each problem class. It is obtained as the ratio between the number of problems where the minimum found by QA is a feasible solution, divided by the number of problems with the same number of items.

Conclusion

This paper introduces QAL-BP, a novel quantum formulation based on the Augmented Lagrangian method for efficiently solving the Bin Packing Problem (BPP) using quantum annealing. QAL-BP offers an analytical estimation of model's penalty terms for a specific class of problem instances, eliminating the need for recursive approximation methods to empirically estimate Lagrangian multipliers. This enhancement amplifies the generalizability of our approach to diverse input instances and improves efficiency by reducing the number of QUBO variables compared to alternative quantum formulations.

We demonstrated the effectiveness of our approach by solving larger problem instances than any previous QUBO formulation for the BPP. Additionally, we present the first experimental comparison of classical and quantum solutions for the BPP, validating that QAL-BP is an analytically correct QUBO formulation obviating the need for empirical estimation of penalty terms. Nevertheless, while our implementation on a quantum annealer does not outperform the state-of-the-art classical solver Gurobi, its TTS exhibits efficient scaling as the problem size increases, considering the current limitations of available quantum technology.

However, several limitations and challenges remain. Firstly, the generalizability of our model to generic BPP instances or other combinatorial optimization problems requires further investigation. Secondly, the limited number of qubits on current quantum annealers poses a significant challenge, restricting the size of problem instances that can be effectively solved. Consequently, testing our model on larger instances and evaluating scalability across a wider range of inputs is currently unattainable. Furthermore, noise and errors in the quantum annealer significantly impact the quality of provided solutions, particularly evident when dealing with larger problems, as demonstrated in experimental results compared to simulated annealing.

To address these challenges, future research will explore advanced quantum hardware with improved qubit accuracy and a greater qubits count. Another promising avenue involves investigating hybrid quantum annealing approaches that leverage classical and quantum methods in tandem, facilitating the solution of larger problem sizes beyond the capabilities of current QPUs. These endeavors are critical to further harnessing the potential of quantum computing in combinatorial optimization problems and propelling the field forward.

Data availability

The dataset generated during the current study is available in the GitHub repository, github.com/Lorenz92/QAL-BP.

Code availability

All code to generate the data, figures and analyses in this study is publicly available with detailed information on the implementation via the following repository: https://github.com/Lorenz92/QAL-BP.

References

1. Garey, M. R. & Johnson, D. S. "Strong" np-completeness results: Motivation, examples, and implications. *J. ACM* **25**, 499–508. https://doi.org/10.1145/322077.322090 (1978).
2. Delorme, M., Iori, M. & Martello, S. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *Eur. J. Oper. Res.* **255**, 1–20. https://doi.org/10.1016/j.ejor.2016.04.030 (2016).

3. Venkatesh, S. M., Macaluso, A. & Klusch, M. Gcs-q: Quantum graph coalition structure generation, in *International Conference on Computational Science*, 138–152 (Springer, 2023).
4. Venkatesh, S. M., Macaluso, A. & Klusch, M. Bilp-q: Quantum coalition structure generation, in *Proceedings of the 19th ACM International Conference on Computing Frontiers*, 189–192 (2022).
5. Venkatesh, S. M., Macaluso, A. & Klusch, M. Quacs: Variational quantum algorithm for coalition structure generation in induced subgraph games. Preprint at arXiv:2304.07218 (2023).
6. Macaluso, A., Klusch, M., Lodi, S. & Sartori, C. MAQA: A quantum framework for supervised learning. *Quantum Inf. Process.* **22**, 159 (2023).
7. Macaluso, A., Orazi, F., Klusch, M., Lodi, S. & Sartori, C. A variational algorithm for quantum single layer perceptron, in *International Conference on Machine Learning, Optimization, and Data Science*, 341–356 (Springer, 2022).
8. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm. Preprint at arXiv:1411.4028 (2014).
9. Hestenes, M. R. Multiplier and gradient methods. *J. Optim. Theory Appl.* **4**, 303–320. https://doi.org/10.1007/BF00927673 (1969).
10. Martello, S. & Toth, P. *Knapsack Problems: Algorithms and Computer Implementations* (Wiley, 1990).
11. Eisemann, K. The trim problem. *Manag. Sci.* **3**, 279–284 (1957).
12. Gilmore, R. & Gomory, R. A linear programming approach to the cutting stock problem I. *Oper. Res.* https://doi.org/10.1287/opre.9.6.849 *(1961)*.
13. Kampke, T. Simulated annealing: Use of new tool in bin packing. *Ann. Oper. Res.* **16**, 327–332. https://doi.org/10.1007/BF02283751 (1988).
14. Loh, K.-H., Golden, B. & Wasil, E. Solving the one-dimensional bin packing problem with a weight annealing heuristic. *Comput. Oper. Res.* **35**, 2283–2291. https://doi.org/10.1016/j.cor.2006.10.021 (2008) (**art Special Issue: Includes selected papers presented at the ECCO '04 European Conference on combinatorial Optimization**).
15. Scholl, A., Klein, R. & Jürgens, C. Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Comput. Oper. Res.* **24**, 627–645 (1997).
16. Vahrenkamp, R. Random search in the one-dimensional cutting stock problem. *Eur. J. Oper. Res.* **95**, 191–200. https://doi.org/10.1016/0377-2217(95)00198-0 (1996).
17. Burke, E., Hyde, M. & Kendall, G. Evolving bin packing heuristics with genetic programming. 860–869. https://doi.org/10.1007/11844297_87 (2006).
18. Falkenauer, E. A hybrid grouping genetic algorithm for bin packing. *J. Heuristics* https://doi.org/10.1007/BF00226291 *(1996)*.
19. Falkenauer, E. & Delchambre, A. A genetic algorithm for bin packing and line balancing, in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, vol. 2, 1186–1192. https://doi.org/10.1109/ROBOT.1992.220088 (1992).
20. Quiroz, M. *et al.* A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Comput. Oper. Res.* **55**, 52–64. https://doi.org/10.1016/j.cor.2014.10.010 (2014).
21. Sim, K. & Hart, E. Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model. https://doi.org/10.1145/2463372.2463555 (2013).
22. Bai, R., Blazewicz, J., Burke, E., Kendall, G. & Mccollum, B. B. A simulated annealing hyper-heuristic methodology for flexible decision support. *4OR* https://doi.org/10.1007/s10288-011-0182-8 *(2012)*.
23. Lopez-Camacho, E., Terashima-Marin, H. & Ross, P. A hyper-heuristic for solving one and two-dimensional bin packing problems, in *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation* (2011).
24. Sosa, A., Terashima-Marín, H., Ortiz-Bayliss, J. C. & Conant-Pablos, S. Grammar-based selection hyper-heuristics for solving irregular bin packing problems. 111–112. https://doi.org/10.1145/2908961.2908970 (2016).
25. Sim, K., Hart, E. & Paechter, B. A hyper-heuristic classifier for one dimensional bin packing problems: Improving classification accuracy by attribute evolution. In *Parallel Problem Solving from Nature - PPSN XII* (ed Coello, C. A. C. *et al.*) 348–357 (Springer, Berlin Heidelberg, 2012).
26. Gomez-Meneses, P. & Randall, M. A hybrid extremal optimisation approach for the bin packing problem. vol. 5865. https://doi.org/10.1007/978-3-642-10427-5_24 (2009).
27. Eilon, S. & Christofides, N. The loading problem. *Manag. Sci.* **17**, 259–268. https://doi.org/10.1287/mnsc.17.5.259 (1971).
28. Gupta, J. N. D. & Ho, J. C. A new heuristic algorithm for the one-dimensional bin-packing problem. *Product. Plan. Control* **10**, 598–603. https://doi.org/10.1080/095372899232894 (1999).
29. Lewis, R. A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing. *Comput. Oper. Res.* **36**, 2295–2310. https://doi.org/10.1016/j.cor.2008.09.004 (2009).
30. Rao, M. R. On the cutting stock problem. *J. Comput. Soc. India* (1976).
31. Dyckhoff, H. A new linear programming approach to the cutting stock problem. *Oper. Res.* **29**, 1092–1104. https://doi.org/10.1287/opre.29.6.1092 (1981).
32. Stadtler, H. A comparison of two optimization procedures for 1- and 1 1/2-dimensional cutting stock problems. *OR Spektrum* **10**, 97–111. https://doi.org/10.1007/BF01720208 (1988).
33. Lodewijks, B. Mapping np-hard and np-complete optimisation problems to quadratic unconstrained binary optimisation problems. https://doi.org/10.48550/ARXIV.1911.08043 (2019).
34. Montanez-Barrera, A., Willsch, D., Maldonado-Romo, A. & Michielsen, K. Unbalanced penalization: A new approach to encode inequality constraints of combinatorial problems for quantum optimization algorithms. https://doi.org/10.48550/ARXIV.2211.13914 (2022).
35. Garcia-de Andoin, M., Oregi, I., Villar-Rodriguez, E., Osaba, E. & Sanz, M. Comparative benchmark of a quantum algorithm for the bin packing problem. https://doi.org/10.48550/ARXIV.2207.07460 (2022).
36. Kuramata, M., Katsuki, R. & Nakata, K. Larger sparse quadratic assignment problem optimization using quantum annealing and a bit-flip heuristic algorithm, in *2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA)*. https://doi.org/10.1109/iciea52957.2021.9436749 (IEEE, 2021).
37. Bertsekas, D. P. *Constrained Optimization and Lagrange Multiplier Methods* (Academic Press, New York, 1982).
38. Lucas, A. Ising formulations of many NP problems. *Front. Phys.* https://doi.org/10.3389/fphy.2014.00005 *(2014)*.
39. Martello, S. & Toth, P. Lower bounds and reduction procedures for the bin packing problem. *Discrete Appl. Math.* **28**, 59–70. https://doi.org/10.1016/0166-218X(90)90094-S (1990).
40. D-wave documentation. https://docs.dwavesys.com/docs/latest/c_qpu_timing.html.
41. Gurobi optimization. Website. Accessed: 22/07/2023.
42. Programming the D-wave QPU: Setting the chain strength. Website. Accessed: 10/11/2023.
43. Carugno, C., Ferrari Dacrema, M. & Cremonesi, P. Evaluating the job shop scheduling problem on a d-wave quantum annealer. *Sci. Rep.* **12**, 6539. https://doi.org/10.1038/s41598-022-10169-0 (2022).
44. Morrison, D. R., Jacobson, S. H., Sauppe, J. J. & Sewell, E. C. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optim.* **19**, 79–102. https://doi.org/10.1016/j.disopt.2016.01.005 (2016).
45. Venkatesh, S. M., Macaluso, A., Nuske, M., Klusch, M. & Dengel, A. Q-seg: Quantum annealing-based unsupervised image segmentation. Preprint at arXiv:2311.12912 (2023).

Author contributions

L.C. developed the theoretical formalism, performed the analytic calculations and numerical experiments. Both A.M. and M.L. supervised the project. L.C. and A.M. wrote the manuscript in consultation with M.L. All authors provided critical feedback and helped shape the research, analysis, and manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to L.C.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.