entropy

Article

Hybrid Classical–Quantum Branch-and-Bound Algorithm for Solving Integer Linear Problems

Claudio Sanavio [1], Edoardo Tignone [2] and Elisa Ercolessi [3,4,*]

[1] Center for Life Nano-Neuroscience at la Sapienza, Fondazione Istituto Italiano di Tecnologia, Viale Regina Elena 291, I-00161 Rome, Italy; claudio.sanavio@iit.it
[2] Leithà S.r.l. | Unipol Group, Via Stalingrado 37, I-40128, Bologna, Italy; edoardo.tignone@leitha.eu
[3] Dipartimento di Fisica e Astronomia "Augusto Righi", Alma Mater Studiorum Università di Bologna, Via Irnerio 46, I-40127 Bologna, Italy
[4] Istituto Nazionale di Fisica Nucleare, Sezione di Bologna,Viale Berti-Pichat 6/2, I-40127 Bologna, Italy
* Correspondence: elisa.ercolessi@unibo.it

Abstract: Quantum annealers are suited to solve several logistic optimization problems expressed in the QUBO formulation. However, the solutions proposed by the quantum annealers are generally not optimal, as thermal noise and other disturbing effects arise when the number of qubits involved in the calculation is too large. In order to deal with this issue, we propose the use of the classical branch-and-bound algorithm, that divides the problem into sub-problems which are described by a lower number of qubits. We analyze the performance of this method on two problems, the knapsack problem and the traveling salesman problem. Our results show the advantages of this method, that balances the number of steps that the algorithm has to make with the amount of error in the solution found by the quantum hardware that the user is willing to risk. The results are obtained using the commercially available quantum hardware D-Wave Advantage, and they outline the strategy for a practical application of the quantum annealers.

Keywords: quantum annealing; binary linear problem; knapsack problem; traveling salesman problem; branch and bound

check for updates

Citation: Sanavio, C.; Tignone, E.; Ercolessi, E. Hybrid Classical–Quantum Branch-and-Bound Algorithm for Solving Integer Linear Problems. *Entropy* **2024**, *26*, 345. https://doi.org/10.3390/e26040345

Academic Editors: Xiao Yuan, Xiaoming Zhang and Bálint Koczor

Received: 10 March 2024
Revised: 10 April 2024
Accepted: 17 April 2024
Published: 19 April 2024

1. Introduction

A Logistic optimization problem, whose goal is to find the solution which minimizes a suitable cost function given a set of constraints, can often be expressed in terms of binary combinatorial problems. One way to tackle this kind of problem consists of exploring all possible solutions, thus pursuing a brute-force strategy. A better strategy is found in the so called branch-and-bound (BB) algorithm [1] that explores sub-combinations of the problem and excludes those that either do not satisfy the constraints or those whose value of the cost function is higher than the solutions previously investigated. Paradigmatic examples that can be solved with the BB algorithm [2–5], are given by (i) The Knapsack Problem (KP) [2], in which one searches for the selection of objects (from a predefined set) that maximizes the load's value while adhering to the capacity constraint of the carrier; (ii) The Traveling Salesman Problem (TSP) [3,6], which aims to find the minimal route that passes through different cities, with the constraint that the path crosses all of the cities exactly once.

In recent times, with physical platforms that have been made available to researchers, attention has been driven by the possibility that quantum computers can speed up the resolution of several NP-hard problems [7]. In this work we analyze a particular quantum computer, the quantum annealer of D-Wave, which is expected to be particularly suitable for solving binary optimization problems. It accomplishes optimization by translating a binary linear optimization problem (BLOP) with constraints into a quadratic unconstrained binary optimization (QUBO) problem. In this way the optimal solution of the BLOP can be described by the ground state of an equivalent Ising problem, which can natively be mapped on the quantum hardware manufactured by D-Wave Quantum Systems Inc. (Burnaby, BC, Canada).

The two problems mentioned above are perfect examples of combinatorial problems that may benefit from the use of a quantum computer. However, few works have been done on the actual resolution of the D-Wave machine. In Ref. [8] the authors tried to solve small KP using the D-Wave quantum annealer, finding results far from the global optimum. In Ref. [9] the authors analyzed the shortest path problem, a slight variant of the TSP. They successfully determined the optimal solution for graphs composed of up to six nodes, a limitation that may not align with current practical requirements. In fact, the quantum annealer is still a developing technology, currently affected by noise that makes it unable to find the global solution even to simple problems.

In order to overcome the apparent failure of the fruitful use of this device in the near period, we propose to use a classical–hybrid protocol, in which the quantum hardware is used as a subroutine for a variant of the classical BB algorithm. In particular, we show that thanks to the BB algorithm we can reduce the size of the problem down to a number of instances that are feasible for the quantum computer. With this strategy, the quantum computer can fully show its potential and find an optimal solution to the problem.
Our findings show that we can exploit near-term quantum computers to speed up the solution of a problem, reducing the number of queries to both the quantum and the classical computer. However, there is a trade off between the quality of the found solution, measured in terms of its proximity to the global optimal solution, and the achievable speed-up. A similar idea has been proposed in Ref. [10], where the BB algorithm was applied to a large number of QUBO problems by simulating the quantum annealer on a classical computer.

In the paper, we apply a hybrid algorithm to study both the KP and the TSP, making use of the real D-Wave machine Advantage [11–13]. The content is as follows. In Section 2.1 we review the definition of the TSP and the KP and we explain how they can be solved using the BB algorithm. In Section 2.4 we show how we can encode the optimal solution of the TSP and the KP as the ground state of a suitable Hamiltonian for the quantum annealer. In Section 3.1 we first discuss the issues related to directly solve the integer linear problems via an unrestricted algorithm on the quantum hardware. Next, we define our hybrid classical–quantum algorithm and study its performance on examples of the KP and TSP problems. In Section 4 we draw our conclusions and present outlooks.

2. Materials and Methods

2.1. The Binary Linear Problem

The BLOP aims to find the minimum of the cost function $z(\mathbf{x})$ over a set of possible solutions Ω, namely

$$\min\nolimits_{\mathbf{x}\in\Omega}\ (\mathbf{x}) = \mathbf{c}^T\mathbf{x} \tag{1a}$$

$$\text{s.t.} \qquad A\mathbf{x} \leq \mathbf{b} \tag{1b}$$

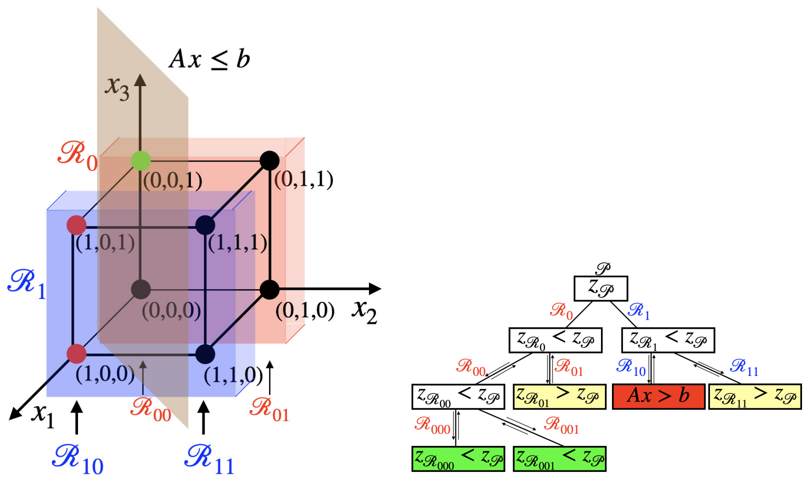
$$\mathbf{x} \in \{0,1\}^N, \tag{1c}$$

with $\mathbf{c}$ being an N-dimensional vector, A being an $m \times N$ matrix, and $\mathbf{b}$ being an m dimensional vector. $\mathbf{x}$ is an N dimensional vector whose components take a value of 0 or 1. Although the simple form of Equation (1), the BLOP is generally NP-hard [14].

In this section we explore one algorithm that is used to solve BLOPs, the BB algorithm. The core procedure of the BB algorithm for finding the solution to the problem $\mathcal{P}_\Omega$ on the set Ω consists of analyzing the restrictions $\mathcal{R}_{\bar{\Omega}}$ of the original set $\bar{\Omega} \subset \Omega$.

The lowest value of the cost function of the original problem $\min_{\mathbf{x}\in\Omega} z(\mathbf{x})$ is a lower bound of the minimum of the cost function of the restricted problem $\min_{\mathbf{x}\in\bar{\Omega}} z(\mathbf{x})$. As we are dealing with a linear problem, the minimum and the maximum of the cost function is to be found at the borders of the set Ω. However the optimal solution could have non-binary values.

Dividing the original problem into sub-problems allows us to consider the restricted problems separately and choose the one with minimum cost. Figure 1a shows the procedure of the BB algorithm when applied to a three-dimensional system. In Figure 1b the procedure is represented as a tree, where each branch represents a restriction of the problem into a selected subset.



(**a**) (**b**)

Figure 1. (**a**) Visualization of the BB algorithm in the three-dimensional space $[0,1]^3$ with constraint $Ax \leq b$. Each restriction $\mathcal{R}$ represents a subset of $[0,1]^3$ where one or more variables have been constrained to a binary value. (**b**) Schematic representation of the BB algorithm as a tree. At each step BB checks if the constraints are satisfied and if the cost function of the node is lower than the current upper bound. When the branching has terminated, the bounding process takes place and the value of the upper bound $z_{\mathcal{P}}$ is updated (green node). If the next branch finds a value of $z > z_{\mathcal{P}}$ (yellow nodes) the branch is not explored further. The same happens if the constraints are violated (red node).

We define $z_{\mathcal{P}}$ as the *current* upper bound of the cost function. We initialize it to the upper bound of the cost function over the set Ω. This is done at the root of the tree and is equivalent to initializing $z_{\mathcal{P}}$ to the value ∞. In the example shown in Figure 1, we analyze the restricted problem $\mathcal{R}_0$, where $\bar{\Omega}$ is a lower dimensional set that has the value of the first variable fixed to zero, i.e., $x_1 = 0$. The lower bound of the cost function in the restricted region is $z_{\mathcal{R}_0} < z_{\mathcal{P}}$, because of the original large value of $z_{\mathcal{P}}$. We then proceed by restricting the second variable $x_2 = 0$, thus defining the problem $\mathcal{R}_{00}$, and finally the third one $x_3 = 0$ in order to specify the problem $\mathcal{R}_{000}$ and find the value of the cost function $z(0,0,0)$ for this restriction. This procedure is called branching and stands for the subsequent restriction of the problem into different sub-problems. Once the value of the cost function for a certain branch has been found, we proceed with the bounding procedure. The current upper bound $z_{\mathcal{P}}$ is updated to $z_{\mathcal{R}_{000}}$. From now on, we consider only subsets where the optimal cost function is less than the updated $z_{\mathcal{P}}$. We now relax the problem, going back to $\mathcal{R}_{00}$, and set $x_3 = 1$. If $z_{\mathcal{R}_{001}} < z_{\mathcal{P}}$, the latter is updated to this value.

We analyze other values of the variables, relaxing the problem and applying different restrictions, as long as we explore all the solutions. If the lower bound of the cost function $z_{\mathcal{R}_i}$ in the follow-up restriction $\mathcal{R}_i$ is greater than the current upper bound $z_{\mathcal{P}}$, we do not need to investigate that restricted region, and we can skip to another branch. If a node in the tree (a restricted subset) does not satisfy the constraints, the node is not considered valid and the bounding process does not take place.

In the tree of Figure 1a each branch develops on the assumption that the value of a variable is fixed to either 0 or 1, but the same method can be applied to integer values of the variables.

Finally, the optimal solution will minimize the cost function while satisfying the constraints.

2.2. The Knapsack Problem

In the KP, we have N objects, each with a value v_i and a weight w_i, for $i = 1, \dots, N$. The problem consists of choosing the items to put in the knapsack so that their total value is maximum while not exceeding the knapsack capacity. This problem can be formulated by using N binary variables x_i ($i = 1, \cdots, N$), with $x_i = 1$ if the object i enters in the knapsack, and $x_i = 0$ if it does not. The values and the weights can be collected in two vectors, $\mathbf{v}$ and $\mathbf{w}$, respectively. The BLOP (1) is written as

$$\min_{\mathbf{x}} \; z(\mathbf{x}) = -\mathbf{v}^T\mathbf{x} \tag{2a}$$
$$\text{s.t.} \quad w(\mathbf{x}) = \mathbf{w}^T\mathbf{x} \leq W, \tag{2b}$$
$$\mathbf{x} \in \{0,1\}^N. \tag{2c}$$

The KP can be solved using the BB algorithm. As we assume that all the weights w_i and the values v_i are positive numbers for all $i = 1, \dots, N$, we can slightly modify the BB algorithm in order to make it more efficient. We call this version KP-BB. We start from the string $\mathbf{x}^0 = (0, \dots, 0)$, which describes the empty knapsack. The value function z and the weight function w are both zero. The first branch has the value of the first variable set, $x_1 = 1$. The value function z is updated to $-v_1$ and the weight function takes value w_1. The k-th branch sets the k-th variable $x_k = 1$ and sets the previous $(k-1)$ variables to zero. At each k-th branch, the value function is $-v_k$ and the weight function is w_k. With this ordering, each of the k branches defines a new knapsack problem, where the k-th object has been chosen and the problem is to choose among the remaining $N-k$ objects. The loading capacity of the new problem is $W - w_k$ and the value of the empty knapsack is initialized to $-v_k$. The KP-BB algorithm recursively applies this restriction of the problem to each branch, following a so-called depth-first search, where the search scouts a tree until it ends, then traces back its steps. At this point, the algorithm passes to another branch. In Figure 2 we show the tree representation of the KP-BB algorithm for a problem with three objects. In the node's box the value of the upper bound of the cost function is shown. When a new solution is explored, the algorithm identifies it as optimal (green), valid but not optimal (yellow), or not valid (red). It stops either when all the variables have been considered, or when the total weight of the knapsack has exceeded its limit.

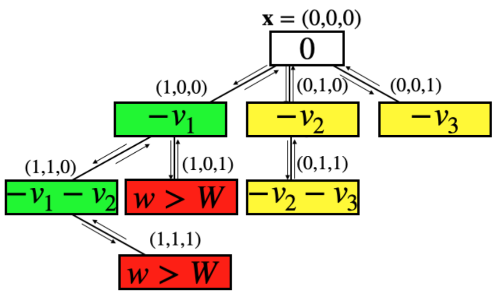


Figure 2. Schematic representation of the KP-BB algorithm as a tree, when applied to a knapsack problem with three objects. The nodes represent a solution as written above them. In the boxes we write the value of the cost function for that solution. The knapsack is initially empty $\mathbf{x} = (0,0,0), z = 0$ and it is filled with the first object, described by the solution vector $(1,0,0)$. The value function is updated to $-v_1$ and the weight to w_1. Hence, a new KP problem is defined as described in the text. A green node represents a valid solution that is currently optimal. A yellow node represents a solution that is valid, but not optimal. A red node represents a non-valid solution, where the constraints are not satisfied.

2.3. The Traveling Salesman Problem

The traveling salesman problem describes an agent that has to visit different cities starting from the depot and returning to it at the end of the journey. The problem consists of finding the route with the lowest cost that visits all the nodes exactly once, where the cost can describe time, distance or money used during the travel. TSP can be modeled by a weighted graph, where the nodes are the cities and the weights of the edges represent the transportation cost from one node to another one. One can always assume the graph to be fully connected, by putting the corresponding cost to be very high (possibly infinity) if two cities are not actually connected by a direct path.

The graph is represented by the pair $G = (V, C)$, with $V = \{d, 1, 2, \ldots, N-1\}$ being the set of N vertices and C being the weighted adjacency $N \times N$ matrix of the graph. The component C_{ij} represents the cost of traveling from node i to node j and it is not necessarily symmetric. A solution for the TSP is the cycle path along the edges of the graph that starts from the depot node, named d, passes through all the other $N-1$ nodes exactly once, and ends in the initial node d. One mathematical description of TSP is given by the Dantzig–Fulkerson–Johnson formulation [15], which considers N^2 binary variables x_{ij}, each representing the edge that connects the node i with the node j. The corresponding linear problem is obtained in the form of Equation (1), collecting the binary variables x_{ij} into a $N \times N$ matrix $\mathbf{x}$.

$$\begin{aligned}
&\min_{\mathbf{x}} \; z(\mathbf{x}) = \mathrm{Tr}[C\mathbf{x}] = \textstyle\sum_{i \neq j} C_{ij} x_{ij}, && \text{(3a)}\\
\text{s.t.} \quad & \textstyle\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i && \text{(3b)}\\
& \textstyle\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j && \text{(3c)}\\
& \textstyle\sum_{i,j \in S} x_{ij} \leq |S| - 1, && \text{(3d)}\\
& \forall S \subset V, \quad 2 \leq |S| \leq N-2, &&\\
& x_{ij} \in \{0, 1\}. && \text{(3e)}
\end{aligned}$$

The first line, Equation (3a) is the objective function that we want to minimize. The other equations are the constraints given by the TSP. In particular, Equations (3b) and (3c) state that each vertex must have one inward and one outward edge, respectively. Equation (3d) avoids the presence of sub-paths that do not cover the whole set of vertices. Here, S is a subset of V and $|S|$ is the number of elements in S. Equation (3e) states that the x_{ij} are binary variables.

The TSP can be solved with the BB algorithm [3,4]. Similar to the knapsack problem, the BB algorithm offers a strategy to systematically explore all solutions, which can be tailored for the specific problem at hand. We refer to this customized version of the branch-and-bound algorithm as TSP-BB.

We start from the depot with initial cost function $z = 0$ and we choose the path to one of the $N-1$ cities, that we denote with $\bar{k}$. Then, the TSP-BB algorithm defines a new TSP made of $N-2$ cities, where the cost matrix is modified to include the previous choice. The new TSP has $N-1$ vertices and new adjacency matrix C' with adjacency components $C'_{i\bar{k}} = C_{id}$. Thus, the initial cost of the new TSP has the updated value $z \to z + C_{d\bar{k}}$. Different from the KP-BB defined before, the TSP-BB has a best-first search approach, where the next node analyzed by the algorithm is the one with the current best z value. When all the variables have been explored in one branch, we define the upper bound of the travel cost z. Any time the initial travel cost of a branch exceeds z, we neglect that branch. On the contrary, we explore the branches with a travel cost lower than z until we find the optimal solution. Figure 3 shows a schematic representation of the TSP-BB algorithm as a tree when applied to a graph with 5 cities.

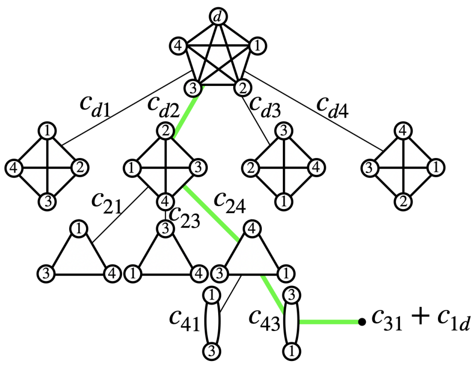


Figure 3. Schematic representation of the TSP-BB algorithm applied to a traveling salesman problem with 5 nodes. Once an edge is chosen, the problem reduces to another TSP with one node less. The initial travel cost is obtained summing up the partial travel costs written on the branches. The travel cost of the colored path is $z = c_{d2} + c_{24} + c_{43} + c_{31} + c_{1d}$. Then, this value is compared to the travel cost of the unexplored branches. If a branch has an initial travel cost lower than the current optimal value, the branch is explored next by the algorithm.

2.4. The QUBO and Ising Formulation

In this work we analyze the use of the D-Wave Advantage quantum computer to solve the combinatorial problems introduced in the previous sections. The D-Wave machine belongs to the class of quantum annealers that work through the application of a global time-dependent Hamiltonian. An introduction to quantum annealers is presented in Appendix A.

The D-Wave machine evolves with an Ising-like Hamiltonian,

$$\begin{aligned} H(s) &= -A(s)\sum_i \hat{\sigma}_x^i \\ &\quad +B(s)\left(\sum_i h_i\hat{\sigma}_z^i + \sum_{i>j} J_{ij}\hat{\sigma}_z^i\hat{\sigma}_z^j\right). \end{aligned} \tag{4}$$

The coefficients $A(s), B(s)$ have the role of the schedule function f in Equation (A2) and s is the adimensional time $s = t/t_a$, normalized with respect to the *annealing time*.

We can write any binary linear problem with constraints as an Ising problem. In fact, any BLOP can be written as a quadratic unconstrained binary optimization (QUBO) problem. This defines a new cost function

$$Q = -\sum_{i=1}^{N+n'} c_i x_i + \sum_{j=1}^{m} \lambda_j (b_j - \sum_{i=1}^{N+n'} A_{ij} x_i)^2, \tag{5}$$

where we have introduced new $n' = \log_2(\max(b_j) + 1)$ binary slack variables that make the problem unconstrained. Each additional i-th slack variable with $i = N+1, \ldots, N+n'$ has cost coefficient $c_i = 0$ and constraint matrix components $A_{ij} = 2^{i-1}/2^N$, with $j = 1, \ldots, m$. The m parameters λ_j are called Lagrange multipliers. Generally speaking, if λ_j are too small, the minimization of the first term is favored, which corresponds to the minimization of the cost function without any constraints. On the other hand, if λ_j is too large, the second term acquires more importance and the optimal solution tends to satisfy the j-th constraint, ignoring the other terms. There is a range of values for each λ_j s.t. and the optimal solution of Q is the optimal solution of Equation (1). However, this range of values is both problem and size dependent.

In order to pass from binary variables into spin variables we define $M = N + n'$ new variables

$$s_i = 2x_i - 1, \quad s_i \in \{-1, 1\}, \tag{6}$$

and set

$$\begin{aligned} h_i &= \frac{c_i}{2} + \sum_{j=1}^{m} \lambda_j b_j A_{ij} - \sum_{j=1}^{m} \frac{\lambda_j}{2} A_{ij} \sum_{k=i}^{M} A_{kj} \\ J_{ij} &= \sum_{k=1}^{m} \frac{\lambda_k}{2} A_{ik} A_{jk}, \end{aligned} \tag{7}$$

that transform the QUBO function Q of Equation (5) into

$$H_Q = \sum_i h_i s_i + \sum_{i>j} J_{ij} s_i s_j, \tag{8}$$

whose minimal energy solution corresponds to the ground state of the quantum Hamiltonian (4) at $t = t_a$.

Although we have a variety of results that show the power of quantun annealers, other results show that when the number of qubits is large the D-Wave machine has difficulty finding the global solution [8]. In the next section we are going to propose a way to circumvent this problem.

3. Results

3.1. Efficient Use of Quantum Annealers in Hybrid Classical–Quantum Algorithm

In this section we propose a hybrid way to use the currently available quantum annealers to produce reliable solution to some NP-hard problems. In Section 2.1 we explained how the BB algorithm can be used to treat either the KP and the TSP. Here, we apply this algorithm to both the problems, stopping when the restricted sub-problems have a size that is small enough that the optimal solution can be obtained by the D-Wave machine.

3.2. The Knapsack Problem

Suppose we want to solve a KP with N objects and capacity W. In order to be able to tract the optimal solution of problem (2), we choose N objects with increasing value $v_i = i$ and with same weight $w_i = 1$ for $i = 1, \ldots, N$. Thus, the optimal solution is the knapsack filled with just the last W objects with the highest value, $\mathbf{x}_{\text{opt}} = (0, \ldots, 0, \mathbf{e}_W)$, with $\mathbf{e}_W$ being the W-dimensional vector of ones, $\mathbf{e}_W = (1, \ldots, 1)$, and with the total value $z_{\text{opt}} = W[N + \frac{1}{2}(1 - W)]$. After including the slack variables, the vector $\mathbf{x}_{\text{opt}}$ has $M = N + \lceil \log_2(W + 1) \rceil$ components.

The QUBO function (5) is

$$Q = -\sum_{i=1}^{M} v_i x_i + \lambda (W - \sum_{i=1}^{M} w_i x_i)^2. \tag{9}$$

Using Equations (2) and (6) we can write the Ising Hamiltonian of the problem. Because Equation (2) has only one constraint, we just need to adjust the one parameter λ. In order to find the optimal value of λ we proceed as follows.

We suppose $\mathbf{y}$ is a solution that satisfies the constraint. If we add a single object, say x_l, that overloads the knapsack, the QUBO function should be s.t.

$$Q(\mathbf{y} + x_l) > Q(\mathbf{y}). \tag{10}$$

As it has to be valid for any possible solution, we need to take the maximum of the right side of Equation (10). Thus, we set $\lambda = \max_i v_i + 1$. This choice ensures that the Hamiltonian ground state is also the optimal solution of the KP.

We have performed an analysis of the bandgap for this particular configuration and we have seen that it scales polynomially with the size of the problem as $M^{-\alpha}$. For some exponent α, see Appendix B. This is a promising feature which means the total annealing time scales as $t_a \sim M^{\alpha}$, and therefore, makes the problem solvable in polynomial time. We have to stress the fact that the polynomial scaling refers to this particular configuration of the problem, and it is not applicable to all the possible KPs.

Let us first examine what happens when we solve the *unrestricted BLOP* on the D-Wave machine. In this case, we find that not only is the solution not the optimal one but also the constraint is not satisfied. Indeed, when the number of qubits is too large, the system is not able to act as an ideal quantum annealer [8], as the results are affected by thermal noise [16]. One may think of increasing the value of the parameter λ in order to force the system to prefer the fulfillment of the constraint. We have analyzed the measured state with lowest energy for different values of λ within the range where the ground state is the optimal solution of our problem, but we have not seen any changes in the outcomes distribution.

Furthermore, we have analyzed how the probability p_0 of measuring the ground state depends on the annealing time, finding it is almost independent of the annealing time, reaching a plateau of the probability. This occurs for any case we consider, including the simple case $W = N$ when the capacity constraint (but not slack variables) becomes ineffective. This is shown in Figure 4, where we plot p_0, obtained as the frequency of the ground state appearing as the outcome, measured out of 1000 reads for different annealing times t_a, calculated for a KP with N-objects and capacity $W = N$.

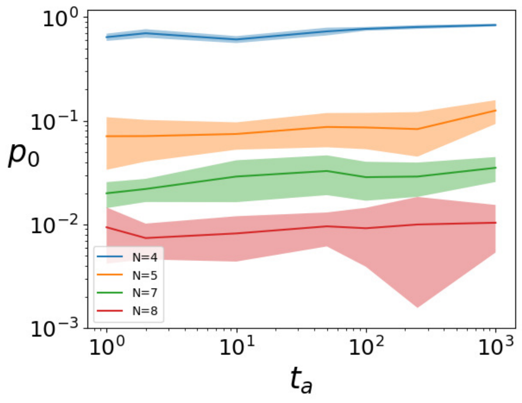


Figure 4. Measured probability p_0 as a function of the annealing time for different N-objects knapsack problems with $W = N$. The value is the average over 20 runs, with 1000 measurements per run.

In Figure 5 we show also the mean value of the minimal energy found on 20 runs of the quantum annealer, each with 1000 measurements. We see that the result is independent of the annealing time. A similar result was found in [9]. The authors analyzed another combinatorial problem, the shortest path problem, with the D-Wave quantum processor and they did not find correspondence between the annealing time and the frequency nor the energy distribution. This means that the behavior of the processor is not dependent on the annealing time, although the theory says otherwise. These results allow us to fix our annealing time to a value of 10 µs.

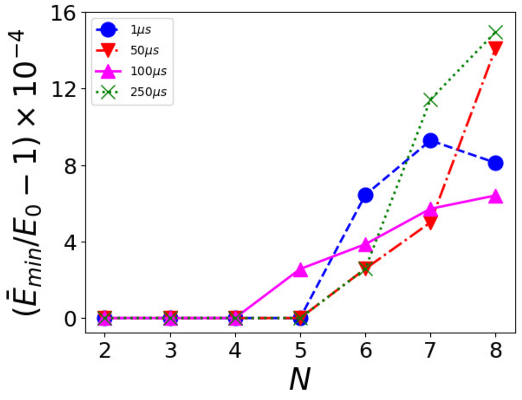


Figure 5. Mean value of the minimal energy found for the N-objects knapsack problems with $W = N$ for different annealing times. The value is an average over 20 runs, with 1000 measurements per run.

To better analyze the quality of the results obtained by the quantum annealer, we can introduce the following three figures of merit that estimate the quality of the obtained solution $\mathbf{x}_a$ with respect to the optimal one $\mathbf{x}_{\text{opt}}$:

– The normalized knapsack value distance

$$\Delta\tilde{v} = \frac{z(\mathbf{x}_a) - z(\mathbf{x}_{\text{opt}})}{z(\mathbf{x}_{\text{opt}})}; \tag{11}$$

– The normalized knapsack weight

$$\tilde{w} = \frac{w(\mathbf{x}_a)}{W}; \tag{12}$$

– The Hamming distance H [7]

$$H(\mathbf{x}_a, \mathbf{x}_{\text{opt}}) = \sum_{i=1}^{M} (\mathbf{x}_a^i \oplus \mathbf{x}_{\text{opt}}^i). \tag{13}$$

Where M is the string length of the solutions and $\oplus$ stands for the sum module 2.

Figure 6 shows these metrics as a function of the number of qubits M used by the quantum annealer in the the case $W = N$. The width of the shaded region is the variance calculated over 20 different runs. We see that already for $M > 8$ the result can differ from the optimal solution and have a variance different from zero.

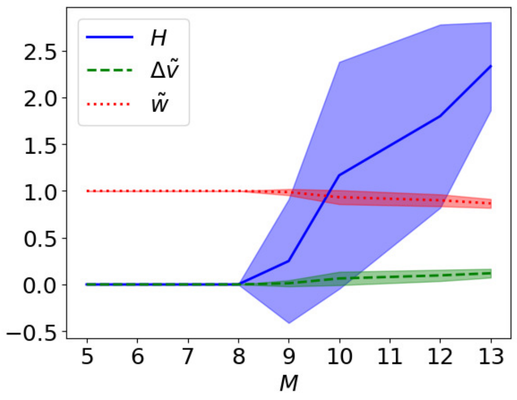


Figure 6. The figure shows $\Delta\tilde{v}$, $\tilde{w}$, and H of Equations (11)–(13), respectively, between the solution found by the quantum annealer $\mathbf{x}_a$ and the optimal solution $\mathbf{x}_{\text{opt}}$. The shaded regions accounts for the variances calculated over 20 runs. In this plot we consider the case $W = N$.

A good trade-off between the certainty of the result and the size of the problem can be provided by the *BB algorithm* that divides the problem into sub-problems that are described by a lower number of qubits, for which the quantum annealer can provide more reliable solutions.

The BB algorithm applied to the knapsack problem, as explained in Section 2.1, explores a number of nodes that go as $\mathcal{O}(2^N)$, since any time they explore a new branch, they create a new KP with one less object and an updated knapsack capacity. In the toy problem we are considering, the number of nodes that saturate the constraint are $\binom{N}{W} = \frac{N!}{W!(N-W)!}$, that is, the number of combinations with W objects chosen from a set of N. This corresponds to the exploration by the KP-BB algorithm of a number of branches

$$n_b = \sum_{k=1}^{W} \binom{N}{k}. \tag{14}$$

This value is lower than the exponential value obtained by applying the BB algorithm depicted in Figure 1 and the reason why is that the optimized KP-BB algorithm makes restrictions on more than one variable at once. But there is no point in making use of a quantum processor, since, for each node, all the variables have been set to a fixed value.

Here, we propose an alternative scheme: we can devise a hybrid classical–quantum protocol by first exploiting the advantage of the BB algorithm to reduce the problem down to a chosen size N', which corresponds to a number of variables $M = N' + \log_2(W' + 1)$ given by the number of remaining objects N' and by the remaining loading capacity of the knapsack W'. Then, we solve the residual problem with the quantum annealer, which might be efficient to solve problems with a relatively small number of qubits; see the diagram in Figure 7.

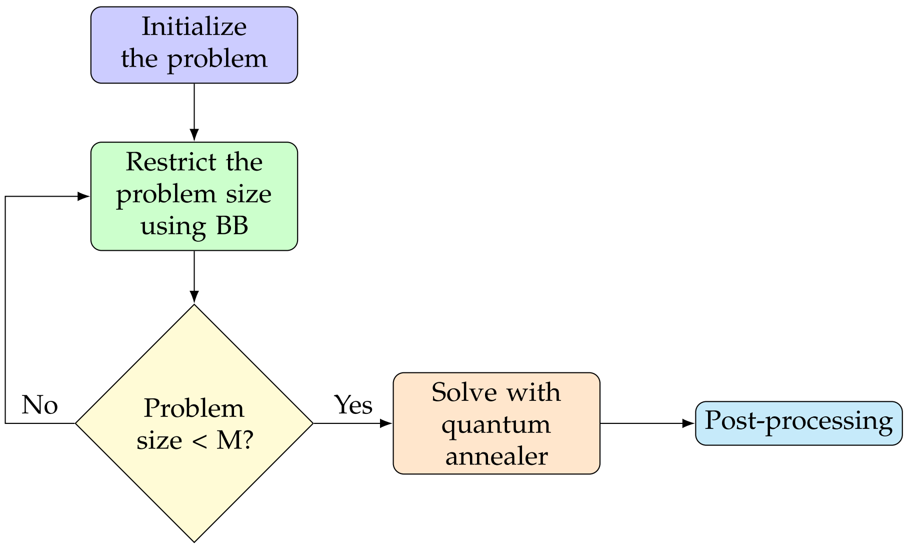

Initialize the problem
Restrict the problem size using BB
No
Problem size < M?
Yes
Solve with quantum annealer
Post-processing

Figure 7. The pseudocode diagram of the hybrid–BB algorithm. We start from the problem $\mathcal{P}$ and we apply the BB algorithm to reduce the problem size down to the chosen number of qubits M. At this point we pass the problem to the quantum annealer. A post-processing step could be required to improve the results.

We consider a KP with $N = 25$ objects and loading capacity $W = 10$. The problem can be described as QUBO by a number of binary variables $M_{\mathcal{P}} = 29$.

Figure 8 shows the number of branches explored by the BB algorithm (green squares) and the number of calls made to the quantum annealer (red circles) as a function of the chosen size M. The horizontal dotted blue line represents the number of branches explored by the optimized KP-BB algorithm, which is constant since it depends on N, W only. We see that when the number of qubits $M > 14$, the number of times the branch-and-bound procedure is applied in the hybrid algorithm is lower than the number of times this is done in the optimized KP-BB algorithm. Each reduced problem is eventually solved by the quantum annealer. In the extreme case of $M = M_{\mathcal{P}}$, the number of branches operated by the classical algorithm is one, as the original problem is defined and promptly sent to the quantum annealer.

Figure 9a–c shows the metrics defined in Equations (11)–(13) for the best obtained solution $\mathbf{x}_a$ as a function of the number of available qubits M for a KP with $N = 25$ and $W = 10$. The width of the shaded region is the variance obtained over 20 different runs. The results are compared with a random outcome of strings, constructed as follows. In order to make a proper comparison, we randomly extract 1000 strings, as many as the number of measurements taken per sample. Then, we choose as our optimal solution the one that has a lower energy, in analogy with the procedure of the D-Wave machine. We then repeat the procedure 1000 times in order to get the statistics.

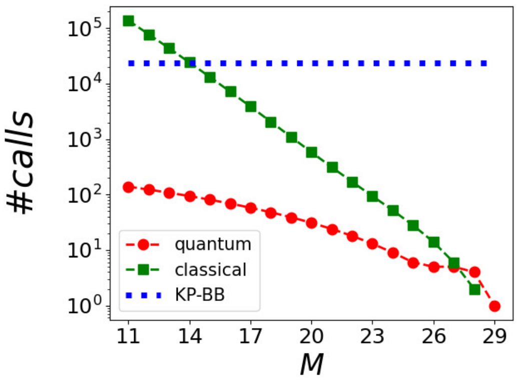


Figure 8. The number of steps of the hybrid–BB algorithm performed on the classical (green squares) and quantum (red circles) computers for a KP with $W = 10$ and $N = 25$ as a function of the maximum number of qubits M used by the quantum annealer. The classical calls count the branching and the bounding procedures, while the quantum calls count the queries to the quantum annealer. This has to be compared with the number of times the fully classical KP-BB algorithm performs branch-and-bound (dotted horizontal blue line).

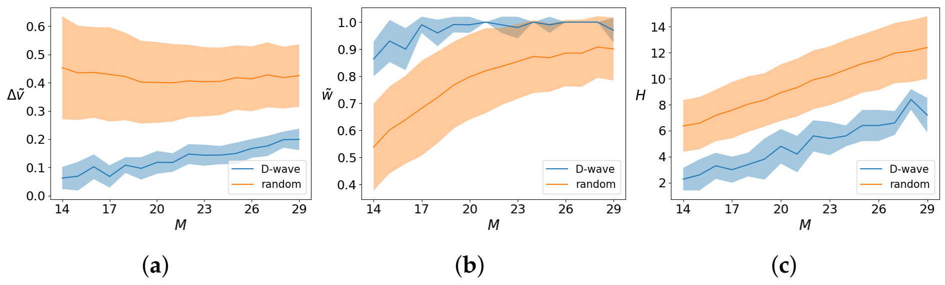


Figure 9. (**a**) The normalized value distance $\Delta\tilde{v}$, (**b**) the normalized knapsack weight $\tilde{w}$, and (**c**) the Hamming distance H between the best solution obtained by the quantum annealer and the optimal solution obtained for a KP with $N = 25$, $W = 10$, solved with the hybrid–BB algorithm with M qubits and averaged over 20 runs. The result is compared to what is obtained by random instances as described in the text.

We see that the outcomes of the quantum annealer are not comparable with random guesses. Figure 9a,b shows that the solution found by the D-Wave machine has a cost value closer to the optimal one and loading closer to the threshold set by the loading capacity. Finally, Figure 9c shows that the Hamming distance is lower than the one reached by random outcomes.

Depending on what is our tolerance with respect to the different quality measures $\Delta\tilde{v}, \tilde{w}$, and H, we may decide to restrict our problem until a certain number of qubits, thus speeding up the resolution of the problem. For instance, we may choose the value $\Delta\tilde{v} = 0.1$ as acceptable, and consider valid any solution that has the constrained satisfied ($\tilde{w} \leq 1$) and a discrepancy between the optimal knapsack value and the obtained value of 10%. In this case, we have analyzed what is the maximum number of qubits $M_{10\%}$ that are necessary to obtain the desired solution. We have kept the knapsack capacity fixed to $W = 10$ and varied the number of items N, thus changing the total size of the problem $M_{\mathcal{P}} = N + 4$. For each $M_{\mathcal{P}}$ we have applied the hybrid–BB algorithm with different values of M. We have then measured 1000 times the system and we have considered only the states for which the constraint was satisfied, discarding the others.

Figure 10a shows the values of $M_{10\%}$ that we obtained for different values of $M_{\mathcal{P}}$. We see that the scaling is almost linear, with a relation $M \sim 0.87 M_{\mathcal{P}}$. The linear scaling is provided by the extra care used to choose the acceptable measurement outcomes. For these cases, we plot in Figure 10b the number of branches generated by the BB algorithm (green squares) and the number of runs of the quantum annealer (red circles). We see that the important variable here is the difference $M_{\mathcal{P}} - M$ that defines the size of the problem tackled by the BB algorithm.

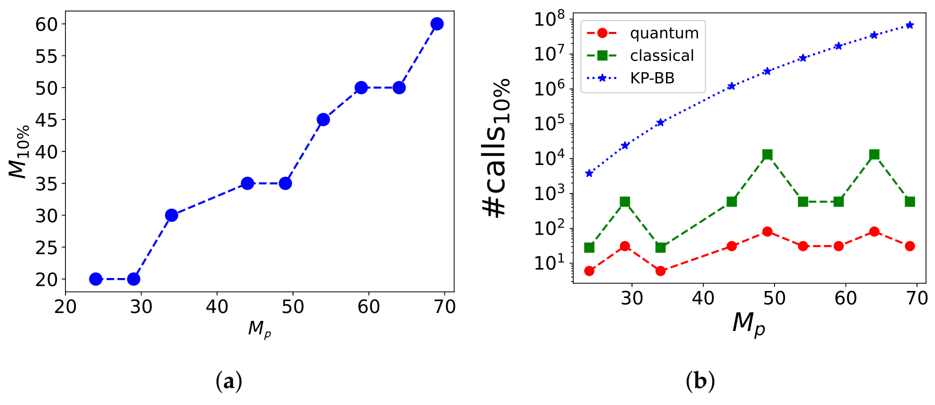


Figure 10. We analyze the resources that are needed to obtain a solution where the constraint is satisfied ($\tilde{w} \leq 1$) and the knapsack value differs by a maximum of 10% from the optimal value ($\Delta\tilde{v} \leq 0.1$). The analysis was performed by keeping $W = 10$ and by varying N. Figure (**a**) shows the maximum number of qubits $M_{10\%}$ for different problem sizes $M_{\mathcal{P}}$. Figure (**b**) shows the number of steps the hybrid–BB algorithm performed on the classical (green squares) and quantum (red circles) computers, to be compared with the number of branches generated by the classical algorithm (blue stars).

Figure 11a–c shows the same experiment performed keeping $W = 10$ and varying the number of objects N and using the maximum number of necessary qubits $M = N + 4$. Although the optimal solution is never reached, the D-Wave machine performs better than random guessing even in this case, as is evident from the results of the metrics. In fact, although the average $\Delta\tilde{v}$ of the D-Wave outcomes is larger then the corresponding value for random guessing, Figure 11a, this is mainly due to the fact that the machine tries to satisfy the capacity constraint, as shown in Figure 11b. Hence, it prioritizes a lower number of objects over the desire for a higher load value. The better performance of the D-Wave machine is captured by the lower value of the Hamming distance in Figure 11c.

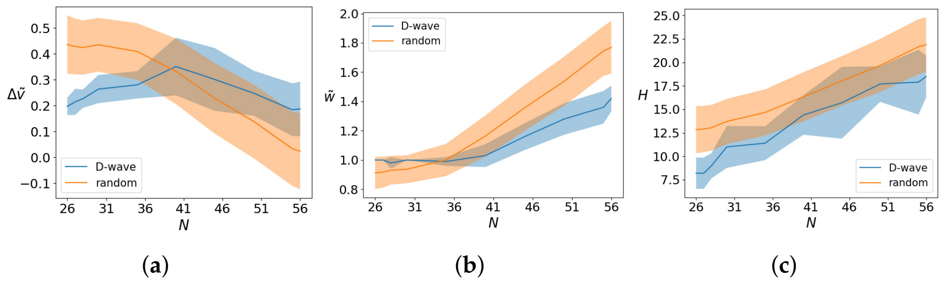


Figure 11. (**a**) The normalized value distance $\Delta\tilde{v}$, (**b**) the normalized knapsack weight $\tilde{w}$, and (**c**) the Hamming distance H between the best solution obtained by the quantum annealer and the optimal solution obtained for a KP with varying N and fixed $W = 10$, completely solved by the quantum processor using $M = M_{\mathcal{P}} = N + 4$ qubits and averaged over 20 runs. The result is compared to what is obtained by random instances as described in the text.

3.3. The Traveling Salesman Problem

Let us move now to solve a TSP with N cities using the quantum annealer, by means of a hybrid classical–quantum protocol similar (but not equal) to the one presented for the KP.

The first step is to write the QUBO formulation of the problem as a function of a N^2 binary vector $\mathbf{x}$ with components $x_{i,j}$, with $i, j = 1, \ldots, N$. We use here the formulation suggested in Ref. [17], such that the first subscript indicates the city and the second indicates the step. Thus, $x_{i,j} = 1$ if the i-th city is visited at step j. Following Equation (3) we can write the QUBO function (5) as [17]

$$\begin{aligned} Q^{\text{TSP}}(\mathbf{x}) \quad = \quad & \sum_{i,j=1}^{N} C_{ij} \sum_{k=1}^{N} x_{i,k} x_{j,k+1} \\ & + \lambda \sum_{i=1}^{N} \left(1 - \sum_{j=1}^{N} x_{i,j}\right)^2 \\ & + \lambda \sum_{j=1}^{N} \left(1 - \sum_{i=1}^{N} x_{i,j}\right)^2, \end{aligned} \tag{15}$$

where the choice $\lambda > \max_{ij} C_{ij}$ ensures that the state with minimal energy is the optimal solution of the TSP.

The TSP-BB algorithm differs from the KP-BB algorithm as it has to cross all the N cities. The maximum number of branches explored by the TSP-BB algorithm is $\sum_{n=1}^{N-1}(N-n)!$. If we decide to stop the BB algorithm when the total size of the problem is M^2 (with M cities left), the number of branches explored by the BB algorithm would be at maximum $\sum_{n=M}^{N-1}(N-n)!$ and the total number of calls to the D-Wave machine would be in the worst case scenario $N!/M!$. However, for the large majority of the problems we have examined, the D-Wave processor is called a very small number of times.

Let us consider a scenario of $N = 10$ cities, all connected together by a route, with a non-symmetric cost matrix given by $C_{ij} = (i - j) \text{mod} N$. The optimal solution $\mathbf{x}_{\text{opt}}$ of this problem has components $x_{ij} = 1$ if $i = j$ and 0 otherwise. This means that the nodes are crossed in the same order that they are labeled.

We have applied the hybrid algorithm to solve this problem and we have obtained a number of calls of the quantum hardware equal to 10 for a number of qubits between 36 ($M = 6$) and 81 ($M = 9$). Clearly these numbers are problem-dependent, but they give a good indication of the small number of calls to the quantum hardware when the hybrid approach is used.

For this problem, as a figure of merit of the quality of the protocol, we now use:

– The travel cost of the found solution $\mathbf{x}_a$, normalized to the travel cost to the optimal solution $\mathbf{x}_{\text{opt}}$:

$$\tilde{c} = \frac{z(\mathbf{x}_a)}{z(\mathbf{x}_{\text{opt}})}; \tag{16}$$

– The Hamming distance $H = H(\mathbf{x}_a, \mathbf{x}_{\text{opt}})$ of Equation (13).

These two metrics are shown in Figure 12a and Figure 12b, respectively, as function of the chosen number of qubits M. As in the previous section, the results are compared with a random path throughout the cities. Contrary to the KP problem, the random solutions can be chosen here to strictly satisfy the constraints, since it suffices to generate the solution with a random ordering of the M cities.

The normalized travel cost is lower than the one obtained for random paths for all the instances analyzed, up to a maximum of 100 qubits ($M = 10$). We see that the optimal solution has been obtained only for the cases with $M = 3, 4$, whereas for larger TSP the D-Wave machine was not able to find it. Also, comparing Figures 12a,b, we see that the Hamming distance alone does not effectively measure the quality of the solution. Already for a TSP with $M = 7$ cities, the results are not distinguishable from random outcomes in terms of H, yet they exhibit a significantly lower travel cost value.

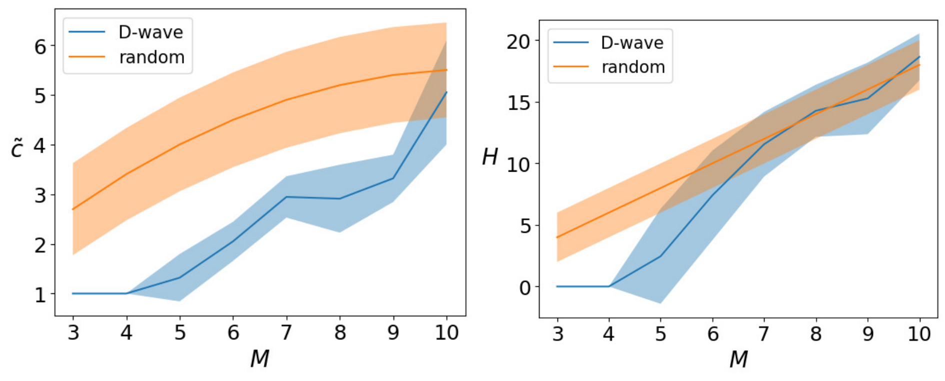


Figure 12. (**a**) The normalized travel cost $\tilde{c}$ and (**b**) the Hamming distance H between the best solution obtained by the quantum annealer and the optimal solution, obtained for a TSP with N cities solved with the hybrid–BB algorithm with M^2 qubits and averaged over 20 runs. The result is compared to what is obtained by random instances as described in the text.

4. Conclusions

In this paper we have investigated the resolution of two binary linear problems, the knapsack problem and the traveling salesman problem, that are known to be NP-hard. These are often solved with the branch-and-bound algorithm that we have described in Section 2.1. After introducing the two problems we have moved our attention to the quantum annealer in Section 2.4, which is a quantum computer that offers a global method for the resolution of binary linear problems. In this work we have merged the classical and quantum method for resolution of these NP-hard problems in order to show the advantage of a hybrid approach. In fact, the branch-and-bound algorithm does not offer a significant speed-up with respect to brute force strategies, whereas quantum annealers suffer from a low reliability when the number of instances is large. However, when merged together, branch-and-bound defines a new problem with a smaller size that can be handled efficiently by the global quantum solver.

Our findings in Section 3.1 show that, despite the annealer itself not demonstrating the theoretical expected sensitivity to variations of the annealing time, its outcomes prove to be significantly better than random, suggesting that a potential advantage can be indeed be obtained within a tailored hybrid framework. Importantly, the hybrid–BB algorithm emerges as a powerful tool which drastically reduces the number of branches generated in comparison with the completely classical counterpart. This, combined with the post-processing step where we select only the solutions that satisfy the constraint, enables us to achieve good results with remarkably few quantum computations (of the order of tens to hundreds). These findings underline the importance of developing hybrid strategies that leverage the strengths of quantum annealers while mitigating their limitations. Further research should explore refined annealing protocols and alternative optimization heuristics to fully unlock the potential of quantum annealing for real-world problems.

Author Contributions: Conceptualization, C.S. and E.T. and E.E.; methodology, C.S. and E.T. and E.E.; software, C.S.; validation, C.S. and E.T. and E.E.; formal analysis, C.S.; investigation, C.S.; resources, C.S.; data curation, C.S.; writing—original draft preparation, C.S.; writing—review and editing, E.T. and E.E.; visualization, C.S.; supervision, E.T. and E.E.; project administration, E.E.; funding acquisition, E.T. and E.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the International Foundation Big Data and Artificial Intelligence for Human Development (IFAB) through the project "Quantum Computing for Applications". E.E. is partially supported by INFN through the project "QUANTUM". C.S. and E.E. acknowledge financial support from the National Centre for HPC, Big Data and Quantum Computing (Spoke 10, CN00000013).

Institutional Review Board Statement: Not applicable

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A. Quantum Annealers

Quantum annealers are a class of quantum computers which make use of a time-dependent evolution Hamiltonian to solve combinatorial problems.

The adiabatic theorem [18,19] is the basis of the computation with quantum annealers. Suppose that a quantum system is subject to a time evolving Hamiltonian $H(t)$, and at time t, the system is in the j-th eigenstate $|\epsilon_j(t)\rangle$, such that $H(t)|\epsilon_j(t)\rangle = \epsilon_j|\epsilon_j(t)\rangle$. The eigenvalues ϵ_j are sorted in increasing order and non-degenerate. The adiabatic theorem states the if the evolution is performed *slowly enough*, the quantum system stays in the evolved j-th instantaneous eigenstate of the Hamiltonian $H(t)$. Understanding what "slowly enough" means is the role of the adiabatic theorem. Several versions of the theorem have been developed during the years, depending on the initial conditions of the system. The Kato version [18] of the adiabatic theorem gives a lower bound for the annealing time t_a in terms of the first and second derivatives of the Hamiltonian $H(t)$, and in terms of the minimum value assumed during the evolution by the energy gap $\Delta_{ij} = |\epsilon_j - \epsilon_i|$, $\forall i \neq j$. An approximate version of the theorem yields the inequality [19]

$$\max_{s\in[0,1]} \frac{|\langle\epsilon_i(s)|\partial_s H(s)|\epsilon_j(s)\rangle|}{\Delta_{ij}(s)^2} \leq t_a, \quad \forall i \neq j. \tag{A1}$$

Since we are interested in the ground state, for our purposes, we consider only the energy gap between the instantaneous ground state and the first energy level $\Delta = |\epsilon_1 - \epsilon_0|$.

The time-dependent Hamiltonian $H(t)$ is composed by two terms, namely [20]

$$H(t) = A(t/t_a)H_0 + B(t/t_a)H_1, \tag{A2}$$

where H_0 and H_1 are two non-commuting Hamiltonians and t_a is the total annealing time. The evolution functions A, B, are such that $B(0) = 0$ and $A(1) = 0$.

The Hamiltonian H_0 has a ground state that is easy to prepare, whereas H_1 is a Hamiltonian for which the ground state corresponds to the classical optimal solution of the combinatorial problem. Thus H_1 is the term that encodes our optimization problem.

At time $t = 0$ the system is prepared in the ground state of the Hamiltonian H_0. For a sufficiently long $t = t_a$, the adiabatic theorem ensures that the state of the system is the solution of our classical problem, that we will recover measuring the system at this time.

The energy gap plays a crucial role in adiabatic quantum computation, and understanding how it scales with problem size is essential. The efficiency of a quantum annealer compared to a classical algorithm largely hinges on the scaling of the band gap. However, it is important to note that the schedule influences the derivative of the Hamiltonian and, consequently, impacts the satisfaction of the inequality (A1).

In order to understand the computational advantage of the quantum annealer we need to define the *quantum speed up*. One definition [19] takes into account the speed up with respect to existing algorithms. Another useful definition is the *limited* quantum speed up [19], which compares a quantum algorithm with its classical analogue, that is an algorithm that proceeds through the same steps in the classical regimes. The classical version of the adiabatic quantum computation is the simulated annealing where the thermal fluctuations allow the exploration of different solutions. Both the classical and quantum annealing algorithms usually find approximations to the global solutions, although we have evidence that quantum annealing reaches the global solutions more often [21] and with shorter annealing time [22–24].

Appendix B. The Energy Gap

In this appendix we consider the Knapsack problem with fixed loading capacity W and increasing number of objects $N \geq W$. As in we did in the main text, all the objects have weight 1 and they are sorted by increasing value $v_i = i$, for $i = 1, \ldots, N$. The number of qubits needed to describe the problem is $M = N + \lceil \log_2(W+1) \rceil$.

In order to be general, we have calculated the minimum value of the energy band gap Δ, considering a one to one embedding into the hardware and considering the schedule functions A, B of Equation (A2) and their magnitudes as provided by D-Wave [25]. Figure A1 shows the minimum of the band gap that is reached during the annealing process. From our simulations, we see that the minimum of the band gap goes as a polynomial of the number of qubits $\Delta E \propto M^{-c}$. The polynomial scaling of ΔE is promising, as it reflects the polynomial scaling of the annealing time needed to find the optimal solution.

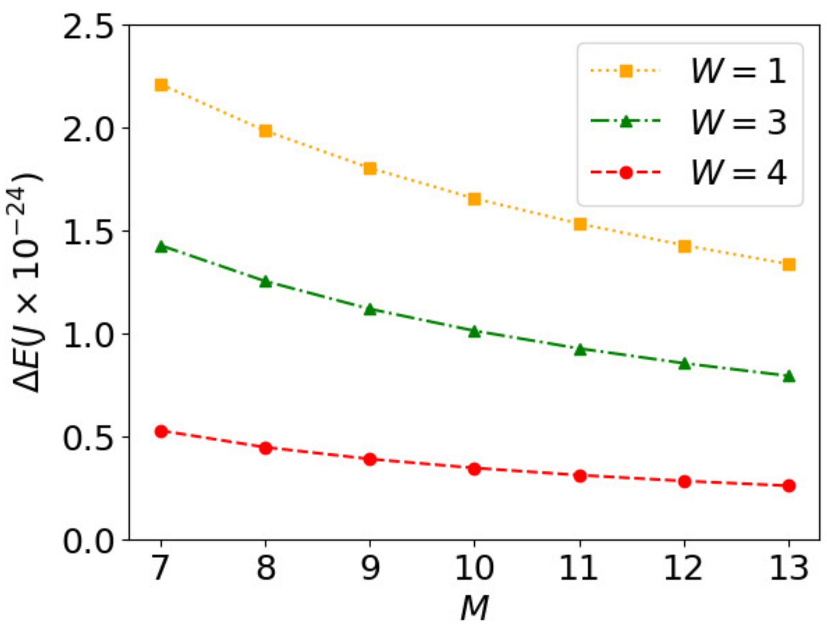


Figure A1. The minimum of the ground state energy band gap ΔE varying the number of qubits M for some KPs with fixed loading capacity W.

However, this result is affected by the topology of the D-Wave quantum annealer. In fact, the results that we found are applicable to an ideal completely connected quantum computer, where Hamiltonian (A2) can be directly implemented. Because of the constrained topology [25] of the machine, several ancillary qubits are used to embed the problem into the quantum annealer, therefore changing the scaling of the band gap. In this work we have used the embedding tool that is implemented in the SDK D-Wave Ocean. Hence we averaged the results over 20 runs, each with a different embedding onto the machine.

Because of the embedding, We can not use the results that we have shown in this appendix to obtain the minimum value of the annealing time through the adiabatic theorem. However, we believe they provide an understanding of the problem and an hardware-agnostic analysis of the energy band gap.

References

1. Land, A.H.; Doig, A.G. An Automatic Method of Solving Discrete Programming Problems. *Econometrica* **1960**, *28*, 497–520. [CrossRef]
2. Martello, S.; Toth, P. *Knapsack Problems: Algorithms and Computer Implementations*; J. Wiley & Sons: Chichester, NY, USA, 1990.
3. Toth, P.; Vigo, D. (Eds.) *The Vehicle Routing Problem*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2002.
4. Laporte, G.; Nobert, Y. A branch and bound algorithm for the capacitated vehicle routing problem. *OR Spektrum* **1983**, *5*, 77–85. [CrossRef]
5. Kolesar, P.J. A Branch and Bound Algorithm for the Knapsack Problem. *Manag. Sci.* **1967**, *13*, 723–735. [CrossRef]
6. Laporte, G. The traveling salesman problem: An overview of exact and approximate algorithms. *Eur. J. Oper. Res.* **1992**, *59*, 2. [CrossRef]
7. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*; Cambridge University Press: Cambridge, UK, 2010.
8. Pusey-Nazzaro, L.; Date, P. Adiabatic Quantum Optimization Fails to Solve the Knapsack Problem. *arXiv* **2020**, arXiv:2008.07456.
9. Krauss, T.; McCollum, J. Solving the Network Shortest Path Problem on a Quantum Annealer. *IEEE Trans. Quantum Eng.* **2020**, 1, 3101512. [CrossRef]
10. Rosenberg, G.; Vazifeh, M.; Woods, B.; Haber, E. Building an iterative heuristic solver for a quantum annealer. *Comput. Optim. Appl.* **2016**, *65*, 845–869. [CrossRef]
11. McGeoch, C.; Farré, P.; Bernoudy, W. D-Wave Hybrid Solver Service + Advantage: Technology Update. In *D-Wave User Manual 09-1109A-V.*; Tech. Rep.; D-Wave Systems Inc.: Burnaby, BC, Canada, 2020.
12. D-Wave Systems. Technical Description of the D-Wave Quantum Processing Unit. In *D-Wave User Manual 09-1109A-V.*; Tech. Rep.; D-Wave Systems Inc.: Burnaby, BC, Canada, 2020.
13. Boothby, K.; Bunyk, P.; Raymond, J.; Roy, A. Next Generation Topology of D-Wave Quantum Processors. *arXiv* **2020**, arXiv:2003.00133.
14. Johnson, D.S. Combinatorial Optimization: Algorithms and Complexity—By Christos H. Papadimitriou and Kenneth Steiglitz. *Am. Math. Mon.* **1984**, *91*, 209–211. [CrossRef]
15. Dantzig, G.; Fulkerson, R.; Johnson, S. Solution of a Large-Scale Traveling-Salesman Problem. *Oper. Res.* **1954**, *2*, 393–410. [CrossRef]
16. Amin, M.H. Searching for quantum speedup in quasistatic quantum annealers. *Phys. Rev. A* **2015**, *92*, 5. [CrossRef]
17. Lucas, A. Ising formulations of many NP problems. *Front. Phys.* **2014**, *2*, 5. [CrossRef]
18. Kato, T. On the Adiabatic Theorem of Quantum Mechanics. *J. Phys. Soc. Jpn.* **1950**, *5*, 435–439. [CrossRef]
19. Albash, T.; Lidar, D.A. Adiabatic Quantum Computing. *Rev. Mod. Phys.* **2018**, *90*, 1. [CrossRef]
20. Annealing Implementation and Controls—D-Wave System Documentation. Available online: https://docs.dwavesys.com/docs/latest/c-qpu-annealing.html (accessed on 5 April 2024).
21. Kadowaki, T.; Nishimori, H. Quantum Annealing in the Transverse Ising Model. *Phys. Rev. E* **1998**, *58*, 5355–5363. [CrossRef]
22. Suzuki, S. A comparison of classical and quantum annealing dynamics. *J. Phys. Conf. Ser.* **2009**, *143*, 012002. [CrossRef]
23. Santoro, G.E.; Tosatti, E. Optimization using quantum mechanics: quantum annealing through adiabatic evolution. *J. Phys. A Math. Theor.* **2008**, *41*, 209801. [CrossRef]
24. Heim, B.; Rønnow, T.F.; Isakov, S.V.; Troyer, M. Quantum versus classical annealing of Ising spin glasses. *Science* **2015**, *348*, 215–217. [CrossRef] [PubMed]
25. Harris, R.; Johnson, M.; Lanting, T.; Berkley, A.; Johansson, J.; Bunyk, P.; Tolkacheva, E.; Ladizinsky, E.; Ladizinsky, N.; Oh, T.; Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor. *Phys. Rev. B* **2010**, *82*, 024511. [CrossRef]