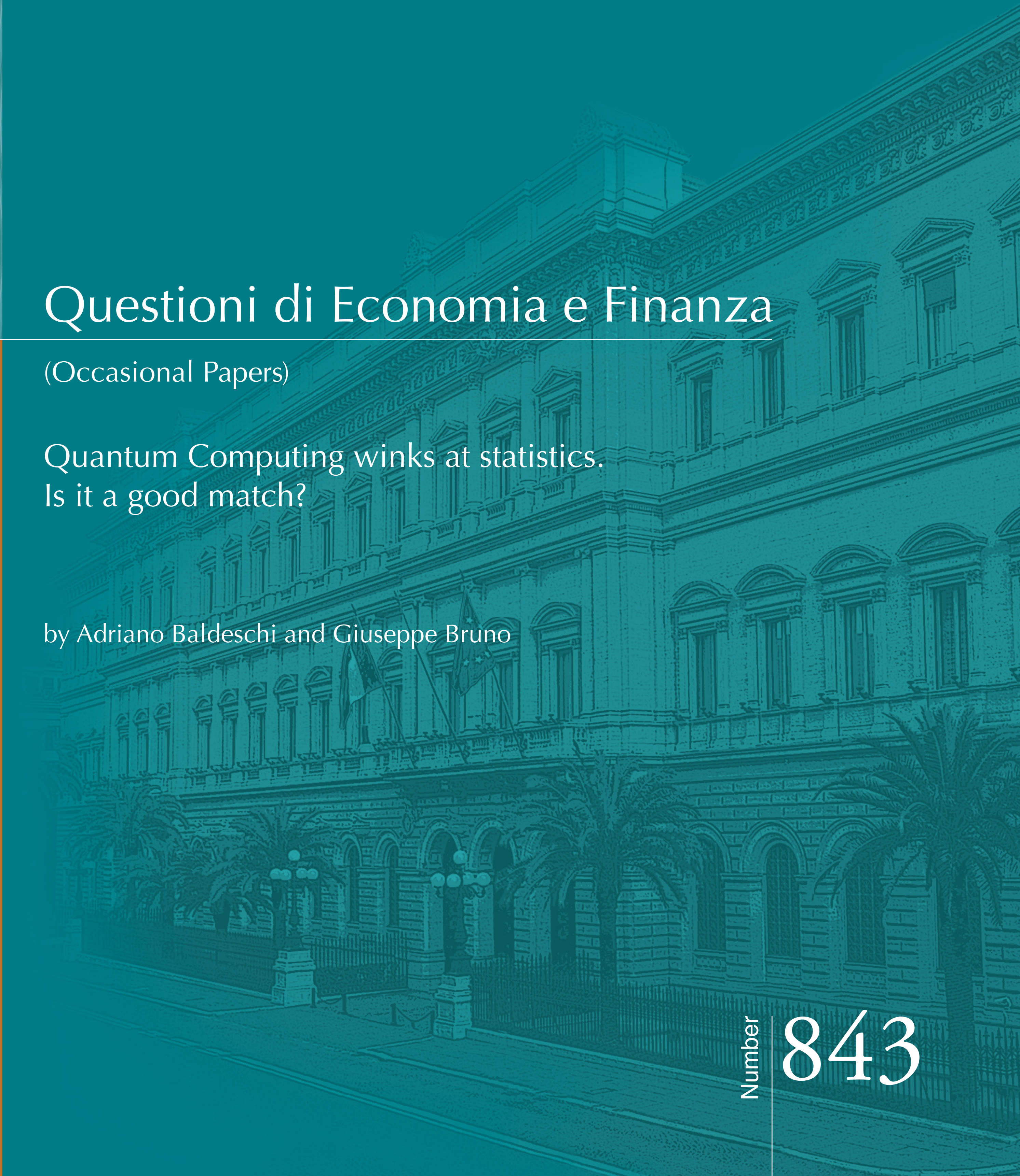
BANCA D'ITALIA
EUROSISTEMA



Questioni di Economia e Finanza

(Occasional Papers)

Quantum Computing winks at statistics. Is it a good match?

by Adriano Baldeschi and Giuseppe Bruno

April 2024

Number 843

BANCA D'ITALIA
EUROSISTEMA

Questioni di Economia e Finanza

(Occasional Papers)

Quantum Computing winks at statistics. Is it a good match?

by Adriano Baldeschi and Giuseppe Bruno

Number 843 – April 2024

ISSN 1972-6643 (online)

Designed by the Printing and Publishing Division of the Bank of Italy

QUANTUM COMPUTING WINKS AT STATISTICS. IS IT A GOOD MATCH?

by Adriano Baldeschi* and Giuseppe Bruno*

Abstract

Quantum computers are widely expected to foster innovation in many fields. Statistical modeling surely has an unquenchable thirst for computing cycles, no matter where they come from. Quantum computing holds great potential to push ahead of its current limits in computational statistics. This paper explores the present suitability of a quantum machine to carry out a basic statistical estimation method, such as the maximum likelihood (ML) estimation for binary discrimination, specifically the Logit model. In this adventure, we encountered different modelling tasks required to map our original mathematical model into a form more suitable to be dealt with by a quantum computer, such as the D-Wave quantum annealer. This platform is specialized for solving Quadratic Unconstrained Binary Optimization (QUBO) problems. Our results show how to leverage these types of computing resources, which appear to be emerging as the main players. The performances and accuracy achieved in this example look quite promising.

JEL Classification: C01, C02, C10.

Keywords: quantum-computing, statistics, econometrics, logit-model.

DOI: 10.32057/0.QEF.2024.0843

---

* Bank of Italy, Economics and Statistics.

1 Introduction

Quantum computing is a field at the intersection of physics, computer science, engineering and mathematics leveraging the capabilities of quantum mechanics (see for examples [15], [11] and [4]).

Quantum computers work in a fundamentally different fashion than classical computers (see [5] and [19]). By leveraging the laws of quantum mechanics, they promise significant speed-ups for particular class of problems. There are many potential applications which include machine learning, new drugs discovery, cryptography simulation (for example see [21], [13] and [3]). This opens up numerous opportunities for the banking industry to harness the advantages of the new technology (see [20]). Computers that use quantum mechanics as their foundation are known as quantum computers. In contrast to traditional computers, which only allow bits to be in one given state at each time: either 0 or 1, quantum computers do computation using devices called *qubits* which may be simultaneously in several states. Due to this characteristic, quantum computers are able to tackle some challenging mathematical problems by reducing their time complexity from exponential to polynomial.

This possibility arises because quantum computers leverage devices called *qubits*: a two-states system which presents itself a blend of superposition of the classical binary states (see [10]).

Unlike conventional computers, quantum machines cannot make use of transistors, logic gates, or integrated circuits. Instead, they utilize *qubits* engineered from elementary particles such as atoms, electrons, photons, and ions, along with information about their spins and states. In theory, quantum computers show more computing power because their *qubits* can be overlapped and operated concurrently.

In this paper we focus our attention on a specific kind of quantum computer, known as the D-Wave[1] quantum annealer. These kinds of machines are not geared for general purpose computation but they are quite good at solving particular optimization problems see ([1]). Quantum annealing processors naturally return low-energy solutions[2].

Physics proves beneficial in addressing optimization by casting it as a problem of energy minimization. A basic principle in physics asserts that objects naturally gravitate towards a state of minimum energy. Whether we have objects descending slopes or hot substances gradually cooling. Quantum annealing harnesses quantum physics to identify low-energy states in a problem, thereby determining the optimal or nearly optimal arrangement of elements (cfr. [8]). The main advantage of the quantum annealer is its ability to solve combinatorial optimization problems much faster than classical computers. This kinds of optimization problems are common in many fields, including finance, logistics, and machine learning, and they often involve finding the best solution out of a large number of possibilities. The reason for focusing on the D-Wave platform

[1]D-Wave is a private company that specializes in quantum computing.

[2]In Economics these might be defined as minimum cost solutions

is two-fold. Firstly, running quantum computation with the D-Wave is much simpler to execute and understand compared to algorithms that operate on universal quantum computers. The second advantage comes from the architecture. Gate model quantum computers typically use a circuit-based architecture, where *qubits* are connected together in a such way that allows them to carry on any kinds of computations. Quantum annealers, on the other hand, employ a lattice-based architecture, where qubits interact with a limited number of the other nearest qubits. This makes quantum annealers simpler and easier to control than gate model quantum computers.

This paper shows the capabilities of a D-Wave quantum machine in finding the optimal parameters to maximize the log-likelihood function of a logit model. The achieved accuracy is on par with that of conventional algorithms. For this goal, we generate synthetic data following a logistic distribution and then transform the problem of maximizing log-likelihood into a Quadratic Unconstrained Binary Optimization (QUBO) problem (see [12]).

The rest of this paper is organized as shown in the following. In section 2 we provide the main taxonomy for the quantum computing. In section 3 we provide the definition of a QUBO problem. In section 4 we give a brief introduction to the D-Wave quantum computers. In section 5 we map the ML problem of a logit model with two parameters into a QUBO. In section 6 we generate the synthetic data following a logistic distribution. In section 7 we show how a D-Wave machine is capable of finding the best parameters for a logit model with one variable and two parameters. In section 8 we compare the results with the ones from classical algorithms. Finally in section 9 we draw some concluding remarks.

2 Main Quantum computing taxonomy

Although the empirical and theoretical efforts of the last three decades, quantum computing is still in its infancy. Today there are quite a few technical challenges that need to be overcome before it becomes a mainstream technology.

These challenges include improving the stability and scalability of quantum hardware, developing better algorithms and error-mitigation and correction techniques, and finding new applications that can take advantage of quantum computing's unique properties [3].

We do have essentially two different flavors of Quantum computing platforms: gate based and quantum annealer. While both kinds of platforms present the potential to solve given particular problems much faster than classical computers, they use different approaches and are best suited for different types of tasks.

These two models are based on different principles and each one has its own strengths and weaknesses.

[3]Quantum error correction entails the detection and correction of errors occurring during computation, while error mitigation provides techniques for compensating the negative effects of these errors.

A gate-based quantum computer implements a universal general purpose platform employing quantum phenomena such as superposition and entanglement to perform any kinds of computations. On the other hand, the quantum annealers are a different class of computers designed specifically for finding the minimum of particular quadratic functions. These devices excel at quickly searching a solution space to find the optimal values. The physical architecture of quantum annealers, such as D-Wave's, employs a network of qubits and couplers among them arranged to efficiently map optimization problems onto the quantum hardware[4]. This layout allows for the effective translation of optimization problems into Hamiltonian functions and their subsequent minimization[5].

The original optimization problem is encoded into the energy levels of a physical system which is left evolving towards the global minimum of the energy landscape. The most common physical implementation of quantum annealers is using superconducting circuits[6]. The system is typically initialized in the ground state of a simple Hamiltonian. Then it slowly [7] evolves cooling down towards the ground state of the final Hamiltonian, which encodes the solution to the problem. A quick grasp of the process can be obtained from figure 1):

Quantum annealing algorithms are based on the adiabatic theorem (see [7]), which states that if the Hamiltonian time evolution is slow enough, the system will remain in the ground state[8] throughout the process. A gate based quantum machine is geared towards the universal computing, which means it can perform any quantum operation represented by a sequence of quantum gates. This makes the gate model quantum computers very flexible and powerful, but also more complex and prone to mistakes. A quantum annealer is a specialized infrastructure aimed at finding the minimum (or maximum) of a function, which represents the objective of our optimization problem. Another difference pertaining the two platforms is their architecture. Gate model quantum computers typically use a circuit-based architecture, where *qubits* are connected together in a way that allows them to perform arbitrary quantum operations. Quantum annealers, on the other hand, use a lattice-based architecture, where qubits are arranged in a lattice and they directly interact only with their neighborhood. This makes quantum annealers simpler and easier to control than their gate-based model counterpart, but also limits their flexibility and the types of problems they can solve.

In terms of performance, gate model quantum computers are generally considered to be more powerful than quantum annealers. This is because they can perform a wider range of quantum operations, which allows them to tackle a

[4]D-Wave employs superconducting qubits as Google, IBM and Rigetti

[5]Hamiltonian functions are also found in dynamic economics. For example when solving for $\max\limits_{w(t),c(t)} \int_0^\infty \exp^{-\rho t} \cdot \frac{c(t)^{1-\theta}}{1-\theta} dt \quad s.t. \quad \dot{w}(t) = w(t) - c(t)$. The Hamiltonian is given by $\mathbf{H} = \exp^{-\rho t} \cdot \frac{c(t)^{1-\theta}}{1-\theta} + \lambda(t) \cdot [w(t) - c(t)]$ where $\lambda(t)$ is the Lagrange multiplier for the law of motion on $w(t)$

[6]Superconductivity is a phenomenon shown by a whole range of materials, that under certain conditions, conduct electrical current without energy dissipation.

[7]here the speed of the evolution heavily depends on the nature of the hardware.

[8]The ground state refers to the minimal energy of the system.

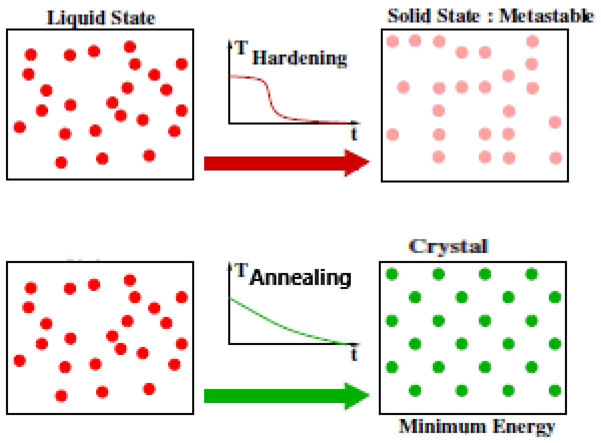


Figure 1: At high temperatures the objects are in a liquid state (left). We can reach the solid state through two main avenues: hardening and annealing. In the hardening process, the material reaches a solid state with non-minimal energy (top right).The atomic structure lacks any symmetry. In the annealing process, the material reaches also a solid state but this time the atoms are symmetrically organized (bottom right). Picture credit [9].

broader range of problems. However, quantum annealers have the advantage of being able to solve certain kinds of combinatorial optimization problems much faster than classical computers, even if they are less powerful than gate model quantum computers.

3 Problem formulation

In many different scientific research areas such as optimization (see e.g.[18]), physics (see, e.g. [2] and combinatorics (see, [14]) we are confronted with QUBO problems. These problems have a wide range of applications. Data science, logistics (quadratic assignment problem), telecommunications (for example frequency assignment problem), portfolio optimization problem, etc. are some examples where solving the underlying quadratic $\{0/1\}$ problems is essential. Formally, a QUBO model or problem is defined as an optimization consisting in looking for the the binary vector q with n_q components which allow to minimize the following objective function:

$$\min_{q_i \in \{0,1\}} E = \sum_i a_i q_i + \sum_{i<j} b_{ij} q_i \cdot q_j, \tag{1}$$

where a_i and b_{ij} are real numbers, and $q_i \in \{0, 1\}$ are the binary decision variables which we have control on. The goal of the problem is to find the values of q_i that minimize equation 1. The computational complexity of a QUBO problem grows exponentially with the number of binary variables as 2^{n_q}. Therefore it quickly becomes unfeasible to find the exact solution in a classical machine when $n_q \geq 30$.

The D-Wave quantum machine leverages the physical phenomenon of quantum annealing to solve any sorts of problems that can be mapped into a QUBO. Equation 1 may be written in a compact way as:

$$\min_{\mathbf{q} \in \{0,1\}} E = \mathbf{q}'\mathbf{Q}x \quad + c' \cdot \mathbf{q} \tag{2}$$

where $\mathbf{q}$ is the n_q dimensional vector composed of the q_i values, $\mathbf{Q} \in \mathbb{R}^{n_q \times n_q}$ is a symmetric matrix and $c \in \mathbb{R}^n_q$. Since $q_i^2 = q_i$ for every $i \in \{1, \ldots, n\}$ one can rewrite equation 2 as $\mathbf{q}'\mathbf{Q}\mathbf{q} \quad + c' \cdot x = q'(\mathbf{Q} + diag(c))x$ where $diag(c)$ is the diagonal matrix whose diagonal elements are given by the entries of the vector c.

The matrix $\mathbf{Q} + diag(c)$ is upper triangular matrix containing the a_i values along the main diagonal and the b_{ij} values in the upper triangle. A QUBO expressed as in equation 2 is the kinds of problems that the D-Wave machine is suitable to solve.

4 The D-Wave quantum machine

A quantum computer is an infrastructure harnessing the characteristics of elementary particles to conduct linear algebra operations in the complex plane without the need to store numerical values in memory and perform computational operations on them. Optimization problems involve the exploration of numerous potential combinations to identify the most favorable outcome. These problems encompass various scenarios, such as resolving scheduling problems like deciding which truck to use for shipping a package or determining the most efficient route for a traveling salesperson to visit multiple cities.

Physics provides a valuable approach to address optimization problems by casting them as challenges of energy minimization. A fundamental principle in physics establishes that systems tend to gravitate toward their state of minimal energy. This rule applies to everyday phenomena, such as objects descending slopes or hot substances gradually cooling. Quantum physics also adheres to this principle, and quantum annealing leverages this principles to determine low-energy states in a problem, thereby revealing the optimal or nearly optimal combination of elements.

While a traditional bit can only represent values as 0 or 1, a qubit is defined as a system existing in the following state:

$$|q\rangle = c_0 |0\rangle + c_1 |1\rangle \tag{3}$$

where $|0\rangle$ and $|1\rangle$ are two orthonormal vectors in the bra/ket notation [9]

[9] The ket or Dirac notations is useful for linear algebra and linear operators on complex vector spaces.

forming a base of a normed vector space. Here c_0 and c_1 are complex numbers satisfying the probability normalization condition $|c_0|^2 + |c_1|^2 = 1$. One of the special characteristics of a qubit consists in the possibility to live simultaneously as a linear combination of the vectors $|0\rangle$ and $|1\rangle$. This connection between the qubit and the traditional bit involves assigning numerical values to the pure states $|0\rangle$ and $|1\rangle$. These values can theoretically vary, but the most common choices are from the sets $\{-1, 1\}$ or $\{0, 1\}$ [10]. While the numerical values of c_0 and c_1 are determined before the qubit is observed, upon observation, the state of the qubit q will collapse into either 0 or 1 with probabilities $P(0) = |c_0|^2$ and $P(1) = |c_1|^2$. While other properties of quantum mechanics are necessary for various quantum computing algorithms, the superposition property is the most important requirement for running a quantum annealing.

Problems described by equation 1) are often solved using methods like simulated annealing (see for example [16]), which allow the exploration both uphill and downhill on the energy landscape to prevent getting stuck in local optimal solutions by employing a stochastic search. In the worst-case scenario, however, these methods may still necessitate the examination of all 2^{n_q} possible combinations of qubit state to identify the lowest-energy solution. Quantum annealing, in theory, offers a way to directly estimate the minimum energy state without the need to explore the entire energy landscape. By maintaining the quantum coherence of the qubit, expressed as a linear combination of the 0 and 1 basis states in equation 3, quantum tunneling becomes possible. Quantum tunneling allows the solution to traverse directly through energy barriers between local minima without the 'retreat' that a method like simulated annealing typically involves.

Once the minimization problem has been mathematically formalized there is the problem of mapping the binary variables onto the physical qubit available. One of the hardest technical challenge is the process of embedding the problem into an actual qubit configuration. While the model has been described as a theoretical graph of qubits, this abstract graph needs to be mapped onto the physical hardware of the D-Wave quantum computer. The hardware graph configurations chosen by D-Wave are known as the Chimera and the Pegasus structure. The two graphs are sparsely connected. Though, in the Chimera graph each qubit can be linked to other 6 qubits whereas in the Pegasus graph each qubit can reach up to 15 qubits. For example, the Chimera structure prevents the direct solution of a simple three-qubit system as there is not a set of three qubits that are all directly connected to each other. To address this, one has to find a way to embed the problem graph onto the Chimera structure to find the minimum energy of the system. This entails linking multiple qubits together in a chain and treating that chain as if it were a single qubit.

Executing a program on the D-Wave quantum computer involves the following steps:

1. Formulate the problem in the form of a QUBO, as shown in equation 1.

[10] It is straightforward to move from the former to the latter representation with the following affine transformation $x_b = \frac{1}{2} \cdot (x_a + 1)$

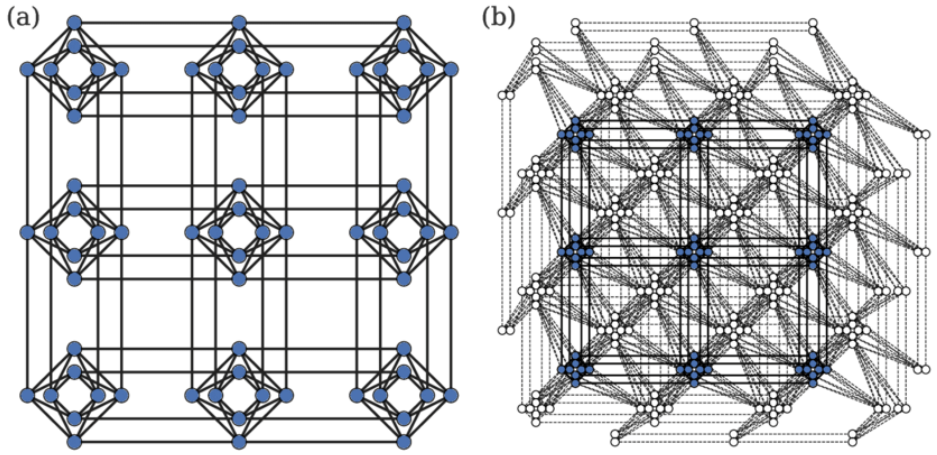


Figure 2: Graphs of Chimera and Pegasus topologies, where nodes and edges represent qubits and couplers, respectively. The picture has been taken from the article [6]).

2. Map the problem's QUBO onto the Chimera or the Pegasus graph. Quite often this step typically requires a conversion from the QUBO into the Ising model which consists going from $\{0, 1\}$ to $\{-1, 1\}$.This conversion can be performed manually or by employing the automated tools provided by D-Wave Ocean software kit [11].
3. Specify the parameters for the annealing process, including factors like the anneal duration or other advanced processing options.
4. Execute the quantum annealing process and collect the results.
5. Reverse the mapping of results from the Chimera or the Pegasus structure to derive a solution and energy for each anneal read. The analysis might involve selecting the solution with the lowest energy or conducting further assessments across all the obtained solutions.

The output of the annealing process using D-Wave quantum computers provides a set of solutions to a given problem, each associated with a specific energy level.

5 Logit model Estimation: a QUBO formulation

In this section we show how to convert the log-likelihood function of a Logit model on a QUBO problem. To keep the calculations within a tractable level, here we consider a logit model with just one feature and two parameters.

[11]Ocean is a suite of open source Python tools.

The logistic distribution and the log-likelihood function for the aforementioned model are respectively:

$$P(y_d = 1, a, b|x_d) = \frac{1}{1 + e^{-a-bx_d}}, \tag{4}$$

and

$$l(a, b|x_d) = -\log(1 + e^{a+bx_d}) + y_d(a + bx_d) \tag{5}$$

respectively, where $x_d \in (x_1, x_1, ..., x_D)$ and $y_d \in \{0, 1\}$ are the data points with sample size D and the labels, respectively. The second order Taylor expansion around $(a, b) = (0, 0)$ of $l(a, b|x_d)$, neglecting the constant terms which are irrelevant in the minimization process, is:

$$l(a, b|x_d) \approx a\left(y_d - \frac{1}{2}\right) + b\left(x_d y_d - \frac{x_d}{2}\right) - ab\left(\frac{x_d}{4}\right) - \frac{a^2}{8} - \frac{b^2 x_d^2}{8}. \tag{6}$$

To simplify the computations, here we consider the expansion around the point $(a, b) = (0, 0)$. In order to be able to take advantage of the D-Wave platform, the log-likelihood function must be expressed in the form of equation 2. Therefore we transform a, b in binary as :

$$a = \sum_{ia=1}^{n_q} 2^{p_{ia}} \cdot q_{ia} \qquad\qquad b = \sum_{ib=1}^{n_q} 2^{p_{ib}} q_{ib}, \tag{7}$$

where p_{ia} and p_{ib} are the powers of 2 to be used for the binary representation of each parameter, n_q is the number of the employed qubits and $q_i \in \{0, 1\}$ are the binary decision variable that we want to find in the optimization process. Note that equation 7 implies $a > 0$ and $b > 0$.

By plugging the parameters a and b from 7 into equation 5 we obtain:

$$l(a, b) = \sum_i \Phi_i q_i + \sum_{j>i} \Theta_{ij} q_i q_j, \tag{8}$$

where

$$\Phi_i = \begin{cases} \sum_d 2^{p_{ia}}\left(y_d - \frac{1}{2}\right) & \text{i qubits of a} \\ \sum_d 2^{p_{ib}}\left(x_d y_d - \frac{x_d}{2}\right) & \text{i qubits of b} \end{cases} \tag{9}$$

and

$$\Theta_{ij} = \begin{cases} \sum_d -\frac{1}{8} 2^{p_{ia}+p_{ja}} & \text{i,j qubits of a} \\ \sum_d -2^{p_{ib}+p_{jb}} \frac{x_d^2}{8} & \text{i,j qubits of b} \\ \sum_d -2^{p_{ia}+p_{jb}} \frac{x_d}{4} & \text{i,j qubits of a and b} \end{cases} \tag{10}$$

Here it is important to remember the fact that $q_i^2 = q_i$. This equivalence show us that equation 8 can easily be mapped as in equation 2 for computational purposes. We should also point that the D-Wave platform finds the minimum of a cost function and hence we have to multiply Φ_i and Θ_{ij} by -1.

6 Data simulation

In section 3 we mapped the bi-dimensional logit model into a QUBO. We now want to test our mathematical formulation of the model with some synthetic data. The procedure to generate data from a logit distribution with one feature and two parameters is trivial:

1. We draw n random points from a uniform distribution with lower bound 0 and upper bound 1. These points (X) are going to be the features.
2. We insert X into equation 4 to obtain the probabilities (p) for the extraction.
3. We finally draw n points from a Bernoulli random variable with probabilities p.

We simulate 10^6 points from equation 4 with $a = 0.3$ and $b = 0.1$. The Python code to extract the data is provided in appendix A. We then chose the values of p_{ia} and p_{ib} to expand in series a and b, respectively and estimate and Θ_{ij} and Φ_i.

Finding the minimum/maximum of equation 8 can be achieved by finding the whole sets of dispositions. It is clear that the whole number of dispositions can not be found when n_q is large since the computational complexity goes as 2^{2n_q}. In Figure 3 we evaluate the cost function for each possible disposition for n_q equal to 3, 5, 7 and 9. The figure reveals that the cost function grows approximately as power-law.

The objective of a good algorithm is to find the disposition associated the the minimum value of the cost function.

7 D-Wave implementation and results

We employed the D-Wave Python library to solve the QUBO problem that represents the log-likelihood maximization for the logistic regression model. This library provides access to quantum annealing hardware, which explores the solution space efficiently to find optimal binary assignments.

After configuring the QUBO problem parameters and submitting it to the quantum annealer, we performed quantum annealing runs to identify the best binary assignment that corresponds to the MLE of the logistic regression model.

As discussed in section 6 we solve the QUBO problem for n_q equal to 3, 5, 7 and 9. The results obtained by the D-Wave library are summarized in table 1 and we compare the results with the optimal values in table 2. The D-Wave Python library successfully found the best solution to our QUBO problem at n_q equal to 3, 5 and 9. When n_q is equal to 9 the values of a and b are 0.2968 and 0.1013, respectively which give a relative error of around 1% ($a = 0.3$ and $b = 0.1$).

In this section, we presented our approach to address a log-likelihood problem for a logit model by mapping it to a QUBO problem and solving it using the

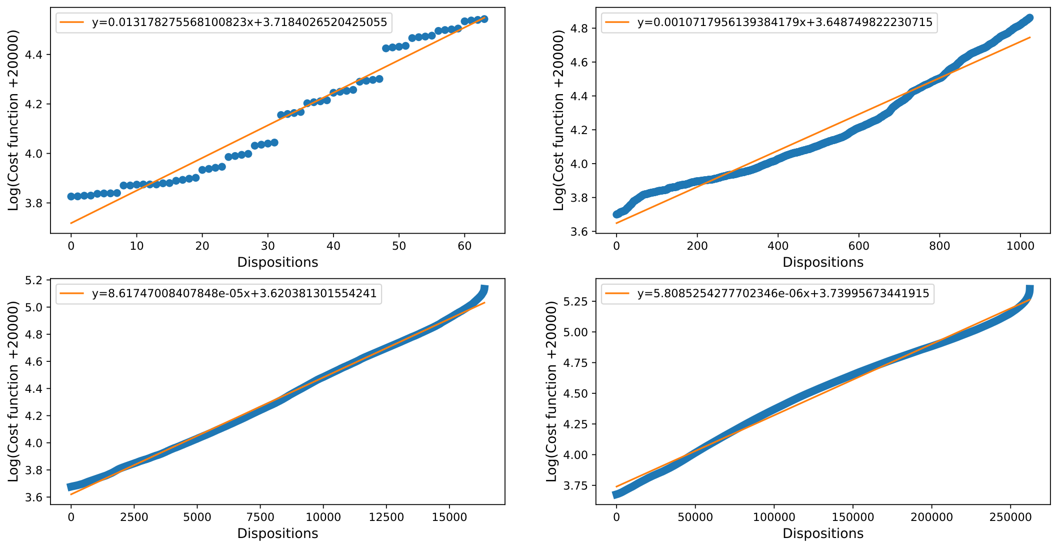


Figure 3: Logarithm of cost function (see equation 8) vs disposition number. We added 20000 to the cost function in order to make the cost function positive for each set of disposition and hence being able to take the logarithm. We evaluate the cost function for each possible disposition of q_a and q_b. Upper left panel: we chose $p_a = p_b = [-1., -4.5, -8]$ therefore there are 64 possible dispositions. Upper right panel: we chose $p_a = p_b = [-1., -2.75, -4.5 - 6.25, -8.]$ therefore there are 1024 possible dispositions. Lower left panel: we chose $p_a = p_b = [-1., -2.17, -3.334, -4.5, -5.667, -6.834 - 8.]$ therefore there are 16384 possible dispositions. Lower right panel: we chose $p_a = p_b = [1., -1.875, -2.75, -3.625, -4.5, -5.375, -6.25, -7.125 - 8.]$ therefore there are 262144 possible dispositions. We also report the best linear fit of the data. Note that that the best linear fit is steeper when n_q is smaller.

n_q	**a**	**b**	**best q_a**	**best q_b**	**Cost function**
3	0.0481	0.5000	[0 1 1]	[1 0 0]	-13301.547
5	0.2099	0.2099	[0 1 1 1 1]	[0 1 1 1 1]	-14993.145
7	0.3504	0.0481	[0 1 1 0 1 1 0]	[0 0 0 1 0 0 1]	-15128.433
9	0.2968	0.1013	[0 1 0 0 0 0 1 1 1]	[0 0 0 1 0 0 1 1 0]	-15247.977

Table 1: Best Parameter values found Using the D-Wave Library for a QUBO Problem. These values should be compared with the optimal values presented in table 2

n_q	**a**	**b**	**best q_a**	**best q_b**	**Cost function**
3	0.0481	0.5000	[0 1 1]	[1 0 0]	-13301.547
5	0.2099	0.2099	[0 1 1 1 1]	[0 1 1 1 1]	-14993.145
7	0.2954	0.1031	[0 1 0 1 1 1 0]	[0 0 1 0 0 0 1]	-15247.918
9	0.2968	0.1013	[0 1 0 0 0 0 1 1 1]	[0 0 0 1 0 0 1 1 0]	-15247.977

Table 2: Optimal QUBO problem parameters were determined by evaluating the cost function for every possible arrangement.

D-Wave Python library. Our results show the potential of quantum annealing as a powerful tool for optimization in statistical modeling, offering a promising avenue for efficient parameter estimation in logistic regression.

8 A comparison with classical algorithms

In this section, we undertake a comprehensive comparison between the solutions of a (QUBO) problems obtained through D-Wave quantum annealing and classical optimization libraries like GEKKO which is an open-source optimization tool designed for solving complex optimization problems in various domains, including engineering, science, and finance. It offers an intuitive and user-friendly interface to define and solve optimization problems, including QUBO problems. GEKKO provides the capability to formulate and solve QUBO problems, which are a class of combinatorial optimization problems. QUBO problems involve binary variables and quadratic objective functions. These problems are encountered in various real-world applications, such as logistics, scheduling, and machine learning. GEKKO is written in Python, which makes it easy to integrate into existing Python-based workflows and applications. It leverages Python's flexibility and extensibility to facilitate problem formulation and solution. GEKKO is equipped with an efficient solver for QUBO problems. It leverages optimization techniques to find the optimal binary assignment that minimizes or maximizes the quadratic objective function. Our aim is to assess the performance and effectiveness of quantum annealing, as offered by D-Wave, in contrast to traditional optimization techniques available in classical libraries. The results obtained with the GEKKO library to our QUBO problem are shown in table 3. The results obtained from D-Wave's quantum computing approach

n_q	**a**	**b**	**best q_a**	**best q_b**	**Cost function**
3	0.0481	0.5000	[0 1 1]	[1 0 0]	-13301.547
5	0.2099	0.1928	[0 1 1 1 1]	[0 1 1 0 0]	-14948.109
7	0.3504	0.0569	[0 1 1 0 1 1 0]	[0 0 0 1 0 1 1]	-15106.367
9	0.2798	0.1727	[0 1 0 0 0 0 0 1 0]	[0 0 1 0 0 1 0 0 0]	-15157.052

Table 3: Best Parameter values found Using the GEKKo Library for a QUBO Problem. These values should be compared with the values obtained with the D-Wave library in table 1

outperform the results obtained from Gekko's library. This implies that for QUBO problems involving a small number of variables, quantum annealing offers a viable solution compared to classical computing platform libraries.

We also employe the scikit-learn library to address a logistic regression (logit) problem and obtain parameter values a=0.2982 and b=0.10687, which are comparable with the values we obtained using the D-Wave library when $n_q = 9$. The scikit-learn library solves the log-likelihood problem directly through numerical methods.

9 Conclusions

This study successfully addressed the solution of maximum likelihood problem for a logit model by mapping it to a Quadratic Unconstrained Binary Optimization (QUBO) problem and solving it using the D-Wave quantum annealer. The utilization of quantum annealing technology provided a novel and efficient approach to finding optimal parameter values. As of today, the size of optimization problems approachable with current annealer technology is limited by the number of qubits and the available connectivity among them. Implementing terms that involve the product of non-adjacent qubits consumes more qubits to represent a single logical variable. Therefore, the current hardware allows solving problems with up to approximately 10 to 15 variables [17].

Furthermore, we conducted a comparative analysis by applying the GEKKO optimization library, a classical solver, to the same problem instances. Our findings revealed that the D-Wave quantum annealer outperformed GEKKO in terms of solution quality. The results showed the superiority of D-Wave's quantum annealing approach for solving QUBO-based likelihood problems. We also utilize the scikit-learn library to address a logistic regression (logit) problem and obtaining comparable results with the values we obtained using the D-Wave library.

This research emphasizes the potential of quantum annealing in the realm of optimization and its application to complex statistical models. While GEKKO remains a valuable tool for various optimization tasks, our study highlights the advantages of quantum computing for specific problem domains, offering effective solutions. These results encourage further exploration of quantum annealing for challenging optimization problems in the future.

References

[1] M. Born and V. Fock. Beweis des Adiabatensatzes. *Zeitschrift für Physik*, 51:165–180, 1928.

[2] F. G. Brandao, Kueng G. P., and França D. S. Faster Qquantum and Classical SDP approximations for Quadratic binary Optimization. *Quantum*, 6:625–645, June 2022.

[3] G. Brassard, Høyer P., Mosca M., and Tapp A. Quantum Amplitude Amplification and Estimation. *Quantum Computation and Information*, 305:35–74, June 2002.

[4] Hughes C., D. Finke, D. German, C. Merzbacher, P. Vora, and H. Lewandowsk. Assessing the needs of the quantum industry. *IEEE Transactions*, 65(2):220–242, June 2022.

[5] Hughes C., D. Finke, D. German, C. Merzbacher, P. Vora, and H. Lewandowsk. Assessing the needs of the quantum industry. *IEEE Transactions*, 65(2):220–242, June 2022.

[6] Calaza Carlos D. Gonzalez, Willsch Dennis, and Michielsen Kristel. Garden Optimization Problems for benchmarking Quantum Annealers. *ArXiv:2001.10827*, pages 1–13, 2021.

[7] Daniel Comparat. General conditions for a quantum adiabatic evolution. *Zeitschrift of Physical Review*, 80, July 2009.

[8] J Copenhaver, Wasserman A, and Wehefritz-Kaufmann1 B. Using Quantum Annealers to calculate ground state properties of molecules. *The Journal of Chemical Physics*, 154:1–26, 1 2021.

[9] D. Delahaye, Chaimatanan Supatcha, and Mongeau Marcel. Simulated Annealing: From basics to Applications. *HAL Open Space*, 2018.

[10] W. Dür and Heusler S. What we can learn about quantum physics from a single qubit. *ArXiv:1312.1463*, pages 1–15, 2014.

[11] Aiello D. Clarice et al. Achieving a Quantum Smart Workforce. *QQuantum Science Technology*, 6(3):1–12, June 2021.

[12] Robert Foster, Weaver Brian, and Gattiker Jim. Applications of Quantum Annealing in Statistics. *ArXiv:1904.06819v2*, pages 1–30, 11 2019.

[13] Lov K. Grover. Fast quantum mechanical algorithm for database search. *Proceedings of 28th ACM Symposium on Theory of Computing*, pages 212–219, 1996.

[14] N. Gusmeroli and Wiegele A. Epedis: an Exact penalty method over discrte sets. *Discrete Optimization*, 44, May 2022.

[15] Meyer J., Passante G., Pollock S., and Wilcox B. Today's interdisciplinary quantum information classroom: Themes from a survey of Quantum Information Science instructors. *Physical Review Physics Education Research*, 13(52):22–51, March 2022.

[16] D. Kirkpatrick, Gelatt C., and Vecchi M. P. Optimization by Simulated Annealing. *IBM Research Report*, 9355, 1982.

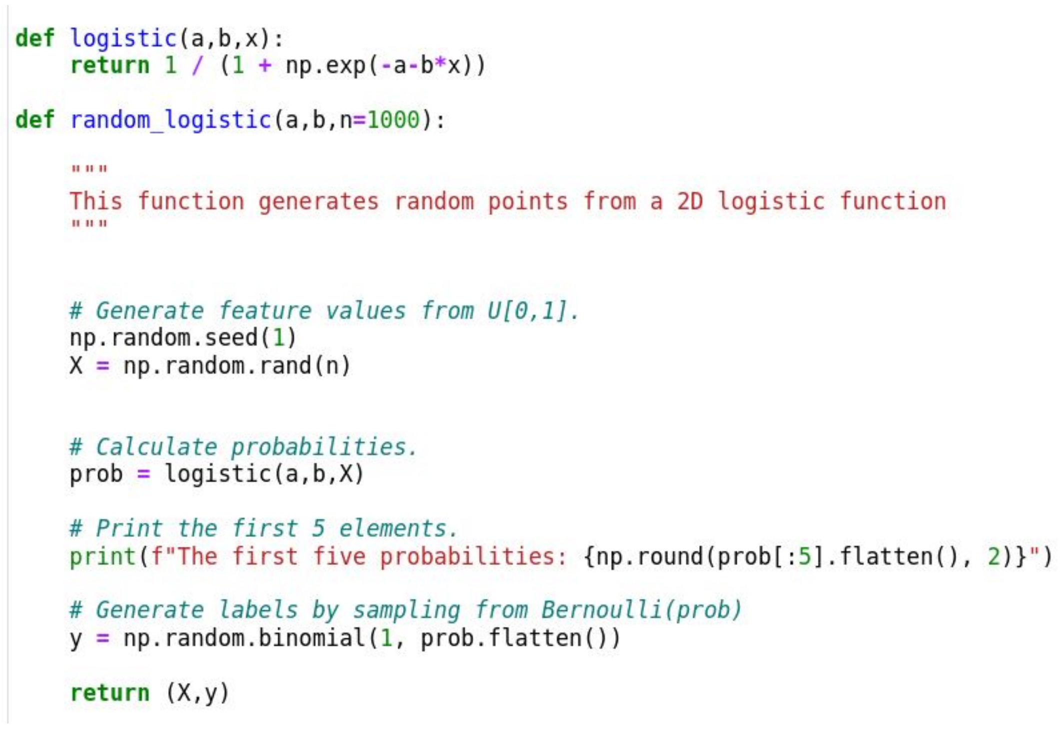
[17] Katzuki R M. Kuramata and Nakata K. Larger Sparse Quadratic Assignment Problem Optimization using quantum annealing and a bit-flip heuristic algorithm. *arXiv:2012.10135v3*, 3, May 2021.

[18] P. M. Pardalos and Rodgers G. P. Computational Aspects of Branch and Bound Aspects of zero-one programming. *Computing*, 45:131–1444, June 1990.

[19] Feynman P. R. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, June 1982.

[20] Wilkens S. and Moorhouse Joe. Quantum Computing for Financial Risk Measurement. *Quantum Information Processing*, 18(1), June 2023.

[21] Peter W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. *Proceedings 35 Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

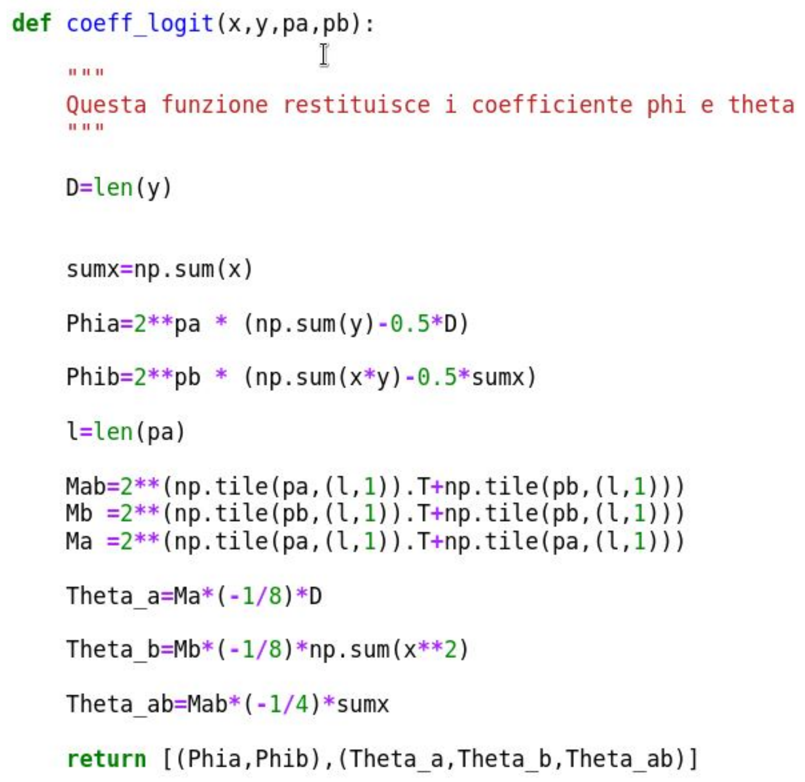


```python
def logistic(a,b,x):
    return 1 / (1 + np.exp(-a-b*x))

def random_logistic(a,b,n=1000):

    """
    This function generates random points from a 2D logistic function
    """


    # Generate feature values from U[0,1].
    np.random.seed(1)
    X = np.random.rand(n)


    # Calculate probabilities.
    prob = logistic(a,b,X)

    # Print the first 5 elements.
    print(f"The first five probabilities: {np.round(prob[:5].flatten(), 2)}")

    # Generate labels by sampling from Bernoulli(prob)
    y = np.random.binomial(1, prob.flatten())

    return (X,y)
```

Figure 4: Python function to produce synthetic data following a logistic distribution with one feature and two parameters.

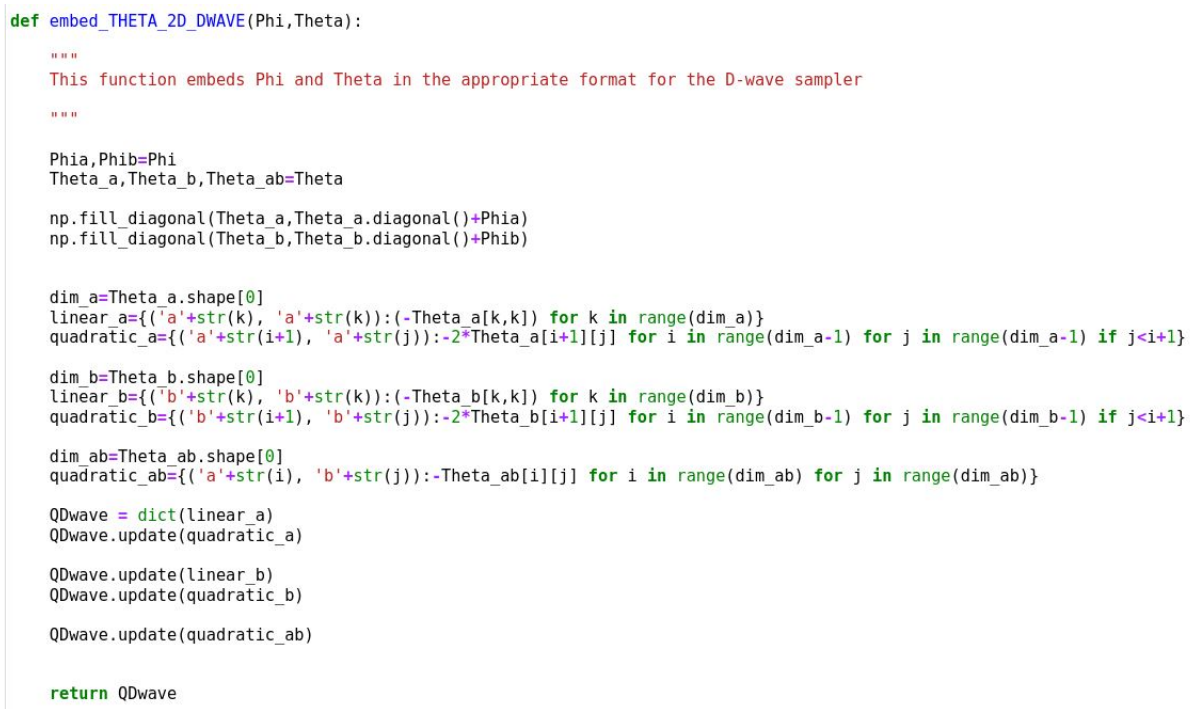
A Python implementation

In this section, we showcase the key Python functions that underpin our work. These functions serve as the backbone of our analysis and enable us 1) to simulate the synthetic data following the logistic distribution (see fig 4), 2) to create the parameters Φ and Θ (see fig 5) and 3) to map Φ and Θ in an appropriate format for the D-Wave machine (see fig 6).



```
def coeff_logit(x,y,pa,pb):
    """
    Questa funzione restituisce i coefficiente phi e theta
    """

    D=len(y)


    sumx=np.sum(x)

    Phia=2**pa * (np.sum(y)-0.5*D)

    Phib=2**pb * (np.sum(x*y)-0.5*sumx)

    l=len(pa)

    Mab=2**(np.tile(pa,(l,1)).T+np.tile(pb,(l,1)))
    Mb =2**(np.tile(pb,(l,1)).T+np.tile(pb,(l,1)))
    Ma =2**(np.tile(pa,(l,1)).T+np.tile(pa,(l,1)))

    Theta_a=Ma*(-1/8)*D

    Theta_b=Mb*(-1/8)*np.sum(x**2)

    Theta_ab=Mab*(-1/4)*sumx

    return [(Phia,Phib),(Theta_a,Theta_b,Theta_ab)]
```

Figure 5: Python function to generate Φ and Θ as in equations 9. and 10.



```python
def embed_THETA_2D_DWAVE(Phi,Theta):

    """
    This function embeds Phi and Theta in the appropriate format for the D-wave sampler

    """

    Phia,Phib=Phi
    Theta_a,Theta_b,Theta_ab=Theta

    np.fill_diagonal(Theta_a,Theta_a.diagonal()+Phia)
    np.fill_diagonal(Theta_b,Theta_b.diagonal()+Phib)


    dim_a=Theta_a.shape[0]
    linear_a={('a'+str(k), 'a'+str(k)):(-Theta_a[k,k]) for k in range(dim_a)}
    quadratic_a={('a'+str(i+1), 'a'+str(j)):-2*Theta_a[i+1][j] for i in range(dim_a-1) for j in range(dim_a-1) if j<i+1}

    dim_b=Theta_b.shape[0]
    linear_b={('b'+str(k), 'b'+str(k)):(-Theta_b[k,k]) for k in range(dim_b)}
    quadratic_b={('b'+str(i+1), 'b'+str(j)):-2*Theta_b[i+1][j] for i in range(dim_b-1) for j in range(dim_b-1) if j<i+1}

    dim_ab=Theta_ab.shape[0]
    quadratic_ab={('a'+str(i), 'b'+str(j)):-Theta_ab[i][j] for i in range(dim_ab) for j in range(dim_ab)}

    QDwave = dict(linear_a)
    QDwave.update(quadratic_a)

    QDwave.update(linear_b)
    QDwave.update(quadratic_b)

    QDwave.update(quadratic_ab)


    return QDwave
```

Figure 6: Python function to map Φ and Θ in an appropriate format for the D-wave machine.