# Virtual Network Function Embedding with Quantum Annealing

1st Pietro Chiavassa
*Politecnico di Torino*
Torino, Italy
pietro.chiavassa@polito.it

2nd Andrea Marchesin
*Politecnico di Torino*
Torino, Italy
andrea.marchesin@polito.it

3rd Ignazio Pedone
*Politecnico di Torino*
Torino, Italy
ignazio.pedone@polito.it

4th Maurizio Ferrari Dacrema
*Politecnico di Milano*
Milano, Italy
maurizio.ferrari@polimi.it

5th Paolo Cremonesi
*Politecnico di Milano*
Milano, Italy
paolo.cremonesi@polimi.it

***Abstract*—In recent years, the growing number of devices connected to the internet led network operators to continuously expand their own infrastructures. In order to simplify this scaling process, the research community is currently investigating the opportunity to move the complexity from a hardware to a software domain, through the introduction of a new paradigm, called Network Functions Virtualisation (NFV). It considers standard hardware platforms where many virtual instances are allocated to implement specific network services. However, despite the theoretical benefits, the mapping of the different virtual instances to the available physical resources represents a complex problem, difficult to be solved classically. The present work proposes a Quadratic Unconstrained Binary Optimisation (QUBO) formulation of this embedding process, exploring the implementation possibilities on D-Wave's Quantum Annealers. Many test cases, with realistic constraints, have been considered to validate and characterise the potential of the model, and the promising results achieved are discussed throughout the document. The technical discussion is enriched with comparisons of the results obtained through heuristic algorithms, highlighting the strengths and the limitations in the resolution of the QUBO formulation proposed on current quantum machines.**

***Index Terms*—Quantum Annealing, Network Functions Virtualisation, Optimisations**

## I. Introduction

Nowadays, the ever-expanding market for devices that require internet access and the distribution of increasingly complex and heterogeneous services have led to substantial efforts in updating the network infrastructures and related management systems. Traditionally, the various functions are performed by special-purpose hardware components, e.g., a firewall, that are located where needed as part of the physical infrastructure. This means that the network functions are rigidly allocated at a hardware level and every update or adjustment to the network requires expensive and time-consuming changes to the physical infrastructure. In order to overcome this limitation, a new paradigm was developed: *Network Functions Virtualisation* (NFV). The aim of NFV is to implement network functionalities with software that can be run and flexibly allocated on general-purpose hardware, therefore removing the need for physical special-purpose devices. Moving the network function allocation from hardware to software allows to drastically improve the flexibility of the network and simplifies its scaling to the new and ever-increasing demands. In particular, network and security functions can be implemented directly via software (e.g., Firewall, IDS, and NAT) in the form of virtual instances, named Virtual Network Functions (VNFs), allocated and executed on traditional physical resources, such as High Volume Servers (HVSs), switches or storage elements. In this context, the problem of finding the optimal mapping of the available resources against certain constraints is known as *VNF Embedding Problem* (VNFEP). A further advantage of Network Functions Virtualisation is the ability to rapidly respond to changes in the network structure, e.g., due to peaks in usage or hardware failures.

Due to the complex nature of the problem, finding an effective solution is challenging as the solution space grows rapidly and exploring it exhaustively is impractical. In previous works, some heuristic strategies have been proposed to obtain good quality solutions in a reasonable time [1]–[4], as a compromise to reduce the computational effort. However, stochastic methods do not guarantee convergence on the optimal solution. Due to the nature of the domain, an ideal approach to solve a VNF embedding problem should be able to find a good solution in a limited amount of time, to ensure it can be used to rapidly respond to changes.

In the last decades, technological advances in the electronics industry have allowed the development of the first quantum computers, with the prospect of increasing the performance of actual processing systems significantly. Although the question of which speedup they offer is still debated, Quantum Annealers shown promising capabilities in effectively tackling tasks such as graph partitioning [5], [6], Support Vector Machines [7], Restricted Boltzmann Machines [8], [9], feature selection [10], [11] and resource allocation [12]. In particular, Quantum Annealers are able to sample low energy solutions of Quadratic Unconstrained Binary Optimisation problems (QUBO).

This paper proposes a QUBO formulation of a general VNF embedding problem which is solved with the D-Wave

978-1-6654-9113-6/22/$31.00 ©2022 IEEE
DOI 10.1109/QCE53715.2022.00048

Advantage quantum annealer. This paper is organised in three stages. First, the feasibility of using Quantum Annealing to solve the proposed VNF embedding problems is analysed. Second, the performance of both quantum and classical solvers (i.e., Tabu search and Simulated Annealing) is compared in different test cases. Finally, the advantages as well as limitations of the Quantum Annealers are discussed.

## II. Background

### A. Network Function Virtualisation

Network and security functions are traditionally implemented through special-purpose software and hardware (e.g., firewall and load balancer), developed in strict combination. Therefore, the functions are not agnostic with respect to the specific physical resources they are running on, making their development less flexible. Network Function Virtualisation proposes to improve the flexibility of the overall system by using standard hardware (e.g., server, switches and storage) to implement these network functions. The physical infrastructure where the VNFs are deployed is typically denoted as *substrate network*.

This approach presents numerous advantages:

- the network infrastructures can be easily scaled up by adding general-purpose servers;
- the different VNFs can be re-allocated easily on different platforms;
- the configuration process can be standardised and centralised, increasing the efficiency for network operators.

The allocation of the Virtual Network Functions on the physical servers is generally done by distributing prebuilt standard software images leveraging the available virtualisation technologies. More details about NFV specifications can be fond in the specification document [13]. Many VNFs can be combined in one or more "chains" of services, namely Service Function Chains (SFCs), to define a more complex network service. The idea is to have a chain of network functionalities between two different endpoints of a distributed network, A and B, where the traffic flows traversing all the elements in the chain. VNFs allow modifying, analysing, and filtering the network traffic.

### B. VNF Embedding Problem

As reported in [14], the VNF embedding problem consists in the mapping of a set of virtual resources (i.e., the VNFs) to a substrate network in which the general-purpose servers are represented as physical nodes. Each of these nodes is described by its own characteristics, resources, and computational capabilities. As an example, a node could be equipped with specific CPU, RAM, and storage resources or have special processing units such as hardware accelerators.

Also, the links connecting the different nodes are characterised by means of specific figures of merit, like bandwidth and delay (e.g., although they may have the same bandwidth, a direct fiber-optic link may have a much lower delay compared to a satellite link). On top of these physical constraints, additional requirements are often introduced, such as the order in which the VNFs within a chain are allocated, or restrictions in hosting VNFs in certain areas of the network. Therefore, during the VNF embedding, all these elements have to be considered in order to perform an optimal mapping. Figure 1 shows an example of problem mapping of a chain composed by 4 VNFs (i.e., Firewall, Load Balancing, Packet Inspection and Reverse Proxy) into a substrate network with 6 physical resources, given their physical connections.

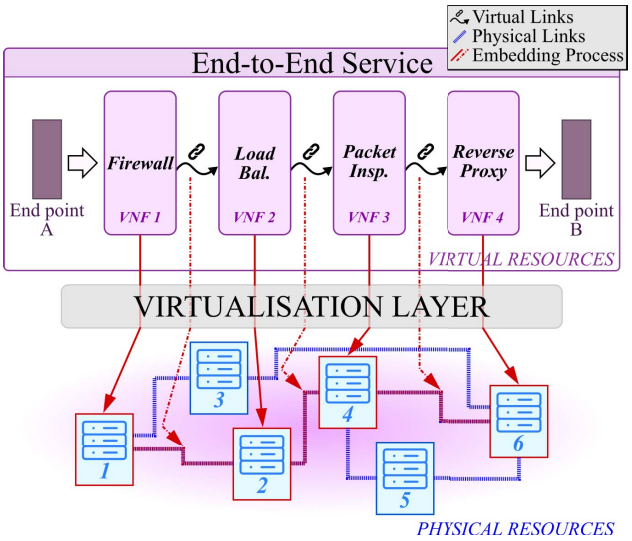


Fig. 1. VNF embedding of a chain of 4 Virtual Network Functions on a substrate network of 6 High Volume Servers, or physical nodes.

## III. Problem Formulation

In the following, a QUBO formulation of the VNF Embedding Problem is proposed.

### A. Substrate Network

The topology of the substrate network is expressed as an undirected graph $G$, comprised of a set of nodes $N$ representing the High Volume Servers (HVS), and a set of edges $E$ representing the physical network channels connecting them. Each server $i \in N$ in the network can provide different types of resources $w \in W$. The quantity of a resource $w$ available on a node $i$ (e.g., CPUs, RAM, storage...) is expressed by $R_i^w$, while its unitary utilisation cost by $C_i^w$. Similarly, each physical network link $(i, j) \in E$, connecting nodes $i \in N$ and $j \in N$, is characterised by its bandwidth $B_{ij}$ and the delay $D_{ij}$ that it introduces. Bandwidth utilisation has a different unitary cost for each link, expressed by $C_{ij}$.

$C_{ij}$ and $C_i^w$ can represent, for example, monetary costs related to energy consumption, infrastructure provider, geographic position and technology generation.

### B. Virtual Network Functions Chains

Each problem defines a set of SFCs, named $P$, which are composed of an ordered list of VNFs. Each SFC $p \in P$ is characterised by two attributes: the maximum allowed delay

$$E_{cost} = \sum_{i,j \neq i} \sum_{p} \sum_{q \neq q_L} \sum_{w} L_{pq}^{ij} \cdot \left( R_{pq}^{w} \cdot C_{i}^{w} + R_{pq+1}^{w} \cdot C_{j}^{w} \cdot \delta_p(q+1) \right) + \sum_{i,j \neq i} \sum_{p} \sum_{q \neq q_L} L_{pq}^{ij} \cdot B_p \cdot C_{ij} \tag{1}$$

for the traffic to go through the chain, indicated with $D_p$, and the maximum bandwidth that it requires, indicated with $B_p$. Each VNF $q$ in chain $p$ requires different resources $w \in W$ from the node on which operates. The quantity of each resource $w$ required by $q$ of chain $p$ is expressed by $R_{pq}^{w}$. In the following, $q+1$ refers to the VNF next to $q$ and $q_L$ refers to the last VNF in the chain to which $q$ belongs.

### C. Problem Variables

A solution to the VNFEP problem should establish which physical nodes host every VNF of every SFC. In order to adhere to the QUBO model, the solution should be expressed in terms of binary variables. In the formulation presented in this work, the chosen binary variables are in the form $L_{pq}^{ij}$. This variable is equal to one if the virtual link between VNF $q$ and VNF $q+1$ of chain $p$ is hosted by the link $(i, j)$ in the direction $i \rightarrow j$. This means that VNF $q$ is placed on node $i$ and VNF $q+1$ on node $j$. The choice of these variables was done to avoid the introduction of terms of order higher than two when expressing the cost function and the constraints.

### D. Cost Function

The quality of a solution to the VNFEP can be expressed in terms of a cost function that evaluates the total cost of node and link resource utilisation required by the VNF embedding. The cost function used in this work is shown in Equation 1 and it is composed of the summation of two terms.

The first term refers to the utilisation cost of node resources required by the VNFs that are allocated. As previously explained, if $L_{pq}^{ij}$ equals to one this means that VNF $q$ of SFC $p$ is allocated on node $i$. If this is the case, the utilisation cost of each resource $w$ required by $q$ is equal to $R_{pq}^{w} \cdot C_{i}^{w}$, hence the product of the resource quantity and the unitary resource utilisation cost. This product alone does not consider the allocation of the last VNF $q_L$ of every chain. For this reason, when VNF $q$ is the second last of chain $p$ and $L_{pq}^{ij}$ equals to one, this means $q_L = q+1$ will be placed on node $j$. Therefore, the cost of this allocation $R_{pq+1}^{w} \cdot C_{j}^{w}$ must be added to the objective function. To achieve this, the Kronecker delta $\delta_p(q+1)$ is equal to one when $q+1$ is the last VNF of the SFC $p$.

The second term, instead, refers to the cost of bandwidth utilisation of the network links. If $L_{pq}^{ij}$ equals to one, the physical link $(i, j)$ carries the traffic from VNF $q$ to VNF $q+1$ of SFC $p$. Since SFC $p$ requires bandwidth $B_p$, the cost of bandwidth utilisation for this traffic on link $(i, j)$ is $B_p \cdot C_{ij}$.

### E. Constraints

In order for a solution to be feasible several constraints must be taken into account, related to both the desired SFC and the resources available on the physical nodes:

- Allocation: each VNF must be allocated exactly on one node. This can bed one easily as shown in Equation 2 by forcing to one the summation of physical links hosting the virtual link between two consecutive VNFs.

$$\forall p, q \neq q_L, \sum_{i,j \neq i} L_{pq}^{ij} = 1 \tag{2}$$

- Continuity: two contiguous VNFs in a SFC must be allocated on nodes that are adjacent in the substrate network. This is expressed in Equation 3 by forcing to zero the difference between the incoming and outgoing contiguous virtual links of the same SFC from a network node ($j$), i.e., for all VNFs except the first and last in the chain, there must always be an incoming and outgoing link.

$$\forall j, p, q \neq q_L, \sum_{i \neq j} L_{pq}^{ij} - \sum_{k \neq j} L_{p,q+1}^{jk} = 0 \tag{3}$$

- Resources: resource utilisation from the VNFs hosted on a node cannot exceed the available resources of the node. In Equation 4, given a node $i$ and a resource $w$, if $L_{pq}^{ij}$ equals to 1, then node $i$ is consuming quantity $R_{pq}^{w}$ of resource $w$ requested by VNF $q$ of SFC $p$. The sum of this contribution from all the VNF on the node is constrained to be lower or equal that the available quantity $R_{i}^{w}$. The second term is only present to consider resource utilisation from the last VNF of the chain.

$$\forall i, w, \sum_{j \neq i} \sum_{p} \sum_{q \neq q_L} L_{pq}^{ij} \cdot R_{pq}^{w} + \sum_{j \neq i} \sum_{p} L_{pq_{L-1}}^{ji} \cdot R_{pq_L}^{w} \leq R_{i}^{w} \tag{4}$$

- Bandwidth: the bandwidth required by the VNFs allocated on a physical link cannot exceed the available one. In Equation 5, given a physical link $(i, j)$, the bandwidth required from all the SFCs traversing the link in both directions is constrained to be lower or equal than the available one.

$$\forall i, j < i, \sum_{p} \sum_{q \neq q_L} (L_{pq}^{ij} + L_{pq}^{ji}) \cdot B_p \leq B_{ij} \tag{5}$$

- Delay: the delay introduced by the physical links cannot exceed the maximum allowed by the chain. In Equation 6, given a chain $p$, the sum of the delay introduced by the physical links traversed by $p$ is constrained to be lower or equal that the maximum allowed delay of $p$.

$$\forall p, \sum_{i,j \neq i} \sum_{q \neq q_L} L_{pq}^{ij} \cdot D_{ij} \leq D_p \tag{6}$$

### F. Full VNFEP QUBO Formulation

The full VNFEP QUBO formulation accounts for both costs and constraints. Since the QUBO formulation does not allow hard constraints, each constraint $v \in V$ must be associated to a penalty $P_v$ to be added to the cost function when a violation occurs.

The formulation of the penalty associated to constraints has been performed according the methodology explained in [15]. The penalty for each equality constraints is therefore proportional to the squared distance from the target value. For inequality constraints, slack variables are introduced.

Finally, each penalty $P_v$ is weighted according to its lagrangian multipliers $\lambda_v$ and added to the cost function (Equation 1). This defines the a final formulation representative of the overall optimisation problem, reported in Equation 7.

$$E_{problem} = E_{cost} + \sum_{v \in V} \lambda_v \cdot P_v \tag{7}$$

### G. Discretization and Slack Variables

Inequality constraints on node resources (Equation 4), link resources (Equation 5) and delay (Equation 6), require the introduction of slack variables in order to be represented. These must be able to assume values ranging from zero to the constraining limit, to avoid the exclusion of feasible solutions.

Since the interval of variation of the slack terms depends on the unit of measure of the specific resource, the influence of the slack terms is discretized by multiplying them for a resource-dependent discretization factor. This allows to reduce the number of slack variables for resources that span larger ranges due to the unit of measure (e.g., RAM size expressed in MB, $10^6$ Bytes). It also permits to even out the number of slack variables required for the different constraints.

An essential aspect is that, if introduced, discretization must be followed not only by slack variables, but also by the constraining factors otherwise the problem formulation becomes inconsistent.

## IV. Methodology

Given the cost function and the constraints of the VNF Embedding Problem (VNFEP) described in Section III, this section describes the methodology that was applied to test the QUBO formulation. The analysis compares the solution quality obtained with various solvers for discrete optimisation problems:

- **Simulated Annealing (SA)**: a known metaheuristic that simulates thermal fluctuations [16]. SA performs a local search for better solutions and stochastically accepts worse solutions with a probability that depends on a temperature parameter that decreases at every step.
- **Tabu Search:** a local search metaheuristic that only accepts worse solutions if no better solution, previously unexplored, is available [17].
- **Quantum Annealing (QA):** a metaheuristic implemented with a physical device that leverages quantum mechanical phenomena to find low energy states. We use in particular the D-Wave Advantage 4.1 Quantum Processing Unit (QPU).

It should be mentioned that in order to use the QPU the first step is to ensure the QUBO problem can fit on its hardware given the limited available qubits and physical connections between them. This can be done via a process called *minor embedding* [18] which transforms the QUBO problem in an equivalent one that accounts for the quantum annealer physical characteristics. In particular we will use the `minorminer` library[1] implementing a heuristic method that runs in polynomial time. An important consequence of the minor embedding process is that a QUBO variable may be represented with a chain of multiple qubits. This is done, for example, if the connections between QUBO variables cannot fit directly on the quantum annealer hardware. Clearly, for a solution to be consistent all qubits representing the same QUBO variable must have the same value. This also means that the number of qubits required by a QUBO problem may be substantially higher than the number of its variables.

In order to avoid ambiguity, we provide the definition of terms used in the testing methodology and result analysis:

- *Chain strength:* the chain strength allows to control how strong is the connection between qubits representing the same QUBO variable. [2]
- *Energy:* the energy of the solution corresponds to the value calculated according to Equation 7 given the corresponding variable values. It is used to assess the quality of a solution, the lower the better.
- *QUBO variables:* the number of QUBO variables required to formulate a specific VNFEP.
- *Qubits:* the number of qubits required by the Quantum Annealer in order to map the QUBO problem to the QPU, resulting from the minor embedding process.
- *Reads:* the number of solutions sampled by a solver. The higher the number of Reads the more likely an optimal solution is found but also the longer the time required by the solver.

In order to measure the computational cost we report the following time measurements:

- *Solver time:* the time required by a solver to find the desired number of solutions, i.e., Reads, as measured by the local client. Since the QPU is accessible via a cloud interface, the QPU solver time only accounts for the actual time required by the QPU, excluding the delay incurred to send the problem via the global network.
- *QUBO time:* the time required to generate the QUBO problem according to the formulation described in section III for a specific VNFEP.
- *QA time:* the annealing time used by the QPU which, together with the number of Reads, increases the prob-

[1] https://docs.ocean.dwavesys.com/projects/minorminer/en/latest/

[2] https://www.dwavesys.com/media/vsufwv1d/14-1041a-a_setting_the_chain_strength.pdf

TABLE I
VNFs AND CONSTRAINT REQUIREMENTS FOR THE SFCs.

| SFC Name | VNFs | Bandwidth (Gbps) | Delay (ms) |
|---|---|---|---|
| `simple secaas` | NAT, Firewall | 1 | 2 |
| `medium secaas` | Firewall, IDS, Reverse proxy | 2 | 3 |
| `complex secaas` | NAT, Firewall, IDS, Reverse proxy, WAF | 3 | 5 |

TABLE II
RESOURCES REQUIRED BY THE AVAILABLE VNFs.

| Resource | Firewall | IDS | NAT | Reverse proxy | WAF |
|---|---|---|---|---|---|
| CPUs | 3 | 4 | 1 | 3 | 3 |
| RAM (GB) | 2 | 4 | 2 | 4 | 3 |

ability of success in finding a good solution. Clearly, increasing either values increases the Solver time.[3]

In general, in our tests, we tune both the number of reads and the chain strength in order to find the solution with the lowest energy value possible, additionally we measure all solver times to compare their performance. It should be noted that the number of SFCs and VNFs is not the only factor that influences the dimension and the complexity of the final VNFEP. Indeed, the network topologies adopted and their related costs and resources are another critical aspect of the problem.

The experiments have all been conducted with an experimental pipeline that is depicted in Figure 2 and is publicly available on GitHub together with additional online material.[4] In detail, the QUBO formulation module is the main one and is in charge of producing a suitable formulation given as input a specific VNFEP (i.e., topology and SFCs to be embedded). After this phase, a set of parameters (e.g., number of reads, chain strength) is provided to the three solvers so that they can be consistently configured. The final results coming from the solvers are then elaborated and stored.

In order to ensure the results are put in the right perspective, for all tests the analysis accounts for the time required to generate the QUBO problem as well as the number of required variables. The analysis also investigates the impact of the problem components, e.g., number of VNFs and SFCs, on the number of variables. This can be used to assess what subset of the networks can fit on the limited qubits available on the QPU. The analysis is performed along two dimensions which are described in the following subsections.

[3]https://docs.dwavesys.com/docs/latest/c_qpu_annealing.html
[4]https://github.com/ptrchv/VNFQuantumOptimization

TABLE III
DEFINED NETWORK TOPOLOGIES.

| | `net0` | `net1` | `net2` | `net3` | `net4` | `net5` |
|---|---|---|---|---|---|---|
| Nodes (#) | 4 | 5 | 5 | 10 | 8 | 7 |
| Edges (#) | 6 | 8 | 8 | 17 | 11 | 12 |

### A. Validation of QUBO Formulation on Classical Solvers

The purpose of this analysis is to validate the QUBO formulation in terms of the computational cost required to generate the QUBO problem and the quality of the solution obtained with several network topologies of growing complexity. For this reason, we randomly generated a set of topologies as Erdős-Rényi graphs of a number of nodes from 4 to 30 and different probabilities of creating an edge.[5] These problems are only solved with classical solvers (Tabu, SA). For this analysis we considered only the constraints component of the VNFEP in order to asses the impact of each constraint on the resulting QUBO problem, on the time required to generate the QUBO model and on the ability of the solver to find a solution that satisfies the constraints. Notice that since this experiment only accounts for the constraints, the best solution will by definition have energy of zero, see Equation 1. Subsequently, the cost components of the VNFEP are included and the resulting solutions obtained with classical solvers are compared.

### B. Quantum Annealing Effectiveness

The purpose if this second analysis is to assess the feasibility of using QA applied to the VNFEP as well as compare the solution quality of QA with the classical solvers to assess its advantages and disadvantages. The experiments are conducted for smaller networks compared to the previous ones due to the current technological limitations of the QPU. The scenario of the tests is the Security-as-a-Service (SECaaS) paradigm, that allows to offer network security services to a customer by using NFV technologies. Consider a security provider that has to offer a service to small and medium-sized enterprises and can choose between three different options: `simple secaas`, `medium secaas`, and `complex secaas`. Each one of these service options corresponds to an SFC that can be deployed on the provider infrastructure and contains a specific number of VNFs as depicted in Table I. More details about the characteristics and resource requirements of those VNFs are available in Table II. The QPU solver has three parameters that affect the quality of the solution, i.e., reads, QA time and chain strength, that need to be selected. The methodology that we adopted is similar to the previous one, the parameters are optimised by accounting first for a simplified formulation only using part of the constraints, then including costs and finally with the full VNFEP formulation. For the QA solver, another aspect arises regarding the *minor embedding* of the QUBO problem on the QPU. As the VNFEP becomes more complex, this process will require a significant amount of time and it is known that the quality of the minor embedding plays an important role in the quality of the solutions. However, the full QUBO problem is highly connected, i.e., most of the quadratic terms are nonzero, this means that it is possible to use a precomputed embedding for a fully-connected QUBO problem of the desired number of variables and the target QPU. This predefined embedding could be, in the future, even made available by the cloud provider that offers access

[5]The *NetworkX* library method `fast_gnp_random_graph` was used.

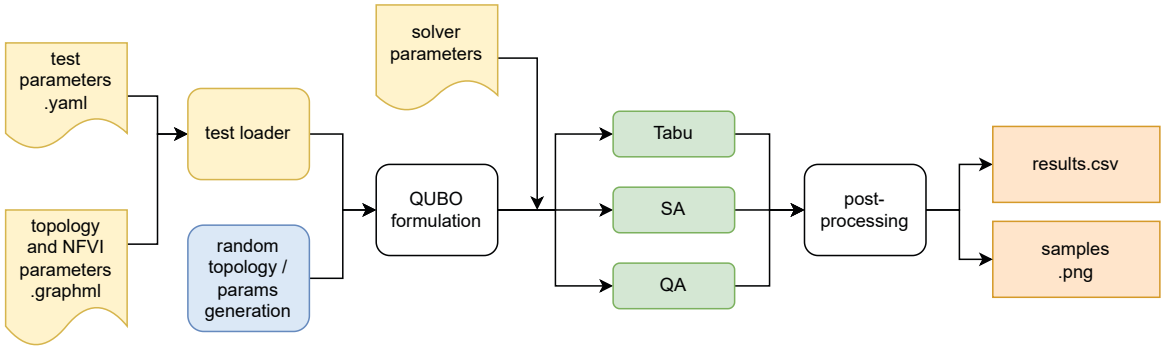


Fig. 2. The workflow above provides a high-level vision of the test and validation process. First, the inputs to the QUBO formulation module are provided: topology (fixed or randomly generated) with its parameters (e.g., node resources) and landscape (e.g., SFCs, VNFs). This stage produces a QUBO formulation to be fed to the different solvers that take as inputs also additional configuration parameters. Finally, the post-processing module elaborates the outcomes of the previous step and produces the final results in the form of images as in Figure 8 and data as in Table IV.

to the QPU. For this reason, the minor embedding time is not considered in the following analysis. In section V, we provide a quantitative analysis of the results. It is also worth mentioning that all the performed tests, the defined topologies, and further analyses are publicly available as part of the online material.

## V. Results and Discussion

This section presents and discusses the results obtained on the two types of analysis described in Section IV.

### A. Validation of QUBO Formulation on Classical Solvers

As described in section IV, this analysis is conducted on a set of randomly generated problems of different sizes and complexity. We first measure the number of QUBO variables depending on the parameters that influence the size and complexity of the VNFEP: the number of nodes within the topology, SFCs, and VNFs. The parameters that impact the most are the number of SFCs and, in particular, the number of VNFs which corresponds to the length of the individual SFC.

As an example, Figure 3 shows the number of QUBO variables required to formulate a VNFEP according to a topology of 20 nodes, by varying the rest of the parameters. It is significant to observe the already large number of variables (2831) required to represent 4 SFCs with 6 VNFs per chain. As a consequence the VNFEP is a challenging problem to represent on the QPU because, after the minor embedding process, it will rapidly require more than the limited number of currently available qubits, i.e., 5000 on the D-Wave Advantage QPU. Figure 4 shows the time required to generate the QUBO problem depending on the number of QUBO variables, compared with the time required by the classical solvers to solve the corresponding problem. While it is expected that the time required to generate the QUBO problem grows with the number of variables, it is significant to observe how its value is comparable to the time required by the classical solvers. Although several factor may affect this comparison, such as differences in how optimised their implementation is,

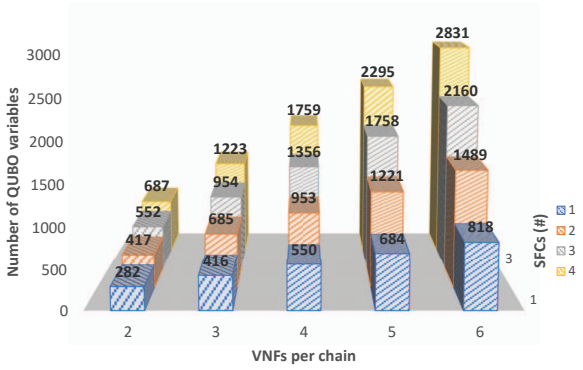


Fig. 3. QUBO variables obtained depending on the number of SFCs and VNFs per chain. The number of nodes is fixed to 20.

this nonetheless indicates how important it is to account not only for the solver time but also for the computational cost required to generate the QUBO problem itself. This mirrors similar findings in [12]. The goal of the first class of tests lies in measuring the solution energy (see Figure 5) and Solver time (see Figure 6) for a network of 20 nodes by varying the number of SFCs and VNFs. As we expected, both energy and solver time increase as the problem becomes more complex. Interestingly, the solution quality of both classical solvers is very close. In Figure 6, it is also apparent the impact of the time required to generate the QUBO problem compared to the solver time.

### B. Quantum Annealing Effectiveness

As described in Section IV, the goal of this analysis is to assess the feasibility of using QA as a solver for the VNFEP with respect to both solution quality, solver time and the size of the problem that can fit the QPU. In order to allow the replication of these findings, all relevant details and

TABLE IV
COMPARISON BETWEEN CLASSICAL SOLVERS AND QPU USING `net3` WITH `simple` AND `medium secaas`.

| Problem | Solver | QUBO variables | Qubits | Solver time (s) | QUBO time (s) | Chain Strength | Lowest Energy | QA time (s) | Reads |
|---|---|---|---|---|---|---|---|---|---|
| Allocation and Continuity Constraints Only | SA | 48 | - | 0,1261 | 0,0036 | - | 0 | - | $10^2$ |
| | Tabu | 48 | - | 2,1120 | 0,0036 | - | 0 | - | $10^2$ |
| | QA | 48 | 138 | 0,2033 | 0,0036 | 150 | 0 | 20 | $10^3$ |
| Allocation and Continuity Constraints with Costs | SA | 48 | - | 0,1187 | 0,0036 | - | 33 | - | $10^2$ |
| | Tabu | 48 | - | 2,1093 | 0,0036 | - | 33 | - | $10^2$ |
| | QA | 48 | 138 | 0,0223 | 0,0036 | 150 | 33 | 20 | $10^2$ |
| Full VNFEP Formulation | SA | 163 | - | 04716 | 0,0437 | - | 69 | - | $10^2$ |
| | Tabu | 163 | - | 2,1797 | 0,0437 | - | 69 | - | $10^2$ |
| | QA | 163 | 2194 | 0,0366 | 0,0437 | 150 | 204 | 40 | $10^2$ |
| | QA | 163 | 2194 | 2,8495 | 0,0437 | 150 | 153 | 50 | $10^4$ |

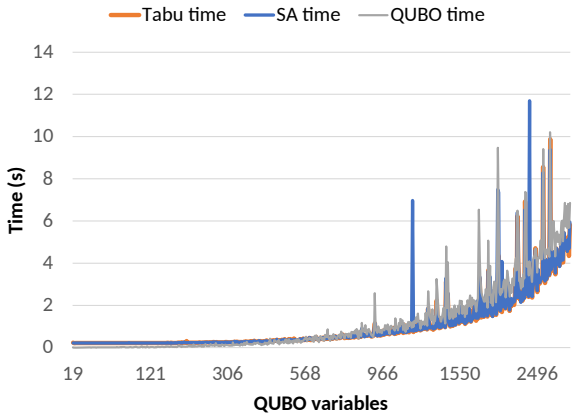


Fig. 4. Tabu Search and SA total time depending on the number of QUBO variables. This test does not apply costs to the objective function but only to the constraints.

configurations, such as the lagrangian multipliers, are available as part of the online material referred in Section IV.

Among the network analysed we focus the discussion on the `net3` topology on which two SFCs are embedded, the `simple secaas` and `medium secaas` described in Table I. The first experiment uses a simplified QUBO formulation that only includes the allocation and continuity constraints, i.e., constraints defined without the use of slack variables, while the costs of the objective function were not considered. The resulting QUBO problem requires 48 variables and 138 qubits after its minor embedding on the QPU. The much larger number of qubits required compared to the QUBO variables is due to the limited connectivity of the available QPU which requires to create numerous qubit chains. As the QPU technology matures and the number of both qubits and connections increases we believe that this issue will be mitigated.

An important aspect to consider is how to set the qubit chain strength. A value that is too low may result in the QA producing inconsistent solutions where qubits associated to the same QUBO variable have different values, while a chain strength that is too high may overwhelm the other components of the energy and result in solutions that are less optimised. Finding the optimal value for the chain strength is still a delicate process. For these experiments the value was chosen starting from the approach suggested in one of the D-Wave white papers[6], which is to use the largest absolute value in the problem's QUBO, and then refining it in a manual process. A similar issue arises with the choice of the lagrangian multipliers required to represent as penalties the constraints required by the VNFEP formulation, see section III.

The results are reported in Table IV. In this case all quantum and classical solvers are able to find solutions of energy zero, meaning that all constraints are satisfied. Comparing the solver times, the slowest solver is the classical Tabu while the fastest is SA. It should be noted however that the QPU is able to explore ten times more solutions, i.e., Reads, than SA only requiring twice the time. The second experiments includes in the QUBO problem the costs. In this case the lowest energy will be higher than zero by definition. Again in Table IV we can see that all solvers find solutions with an energy value of 33. In this case QA is much faster than in the previous one, being significantly faster than all classical solvers. This is due to QA being able to find the best solution when using a much lower number of Reads than in the previous case. The resulting solution can be visualised in Figure 8, with colours indicating that the two SFCs are allocated on three physical nodes.

It should be noted that when only allocation and continuity constraints are enabled, the resulting QUBO problem is composed by a set of smaller independent problems one for each SFC. This can be seen on the left side of Figure 7, the `simple secaas` and `medium secaas` are visible with the `simple secaas` requiring the lowest number of variables. The right side of Figure 7 shows the same VNFEP when the QUBO formulation accounts for all costs and constraints The results show a highly-connected graph where the different SFCs are indistinguishable. The increased connectivity of the

[6]https://www.dwavesys.com/media/vsufwv1d/14-1041a-a_setting_the_chain_strength.pdf

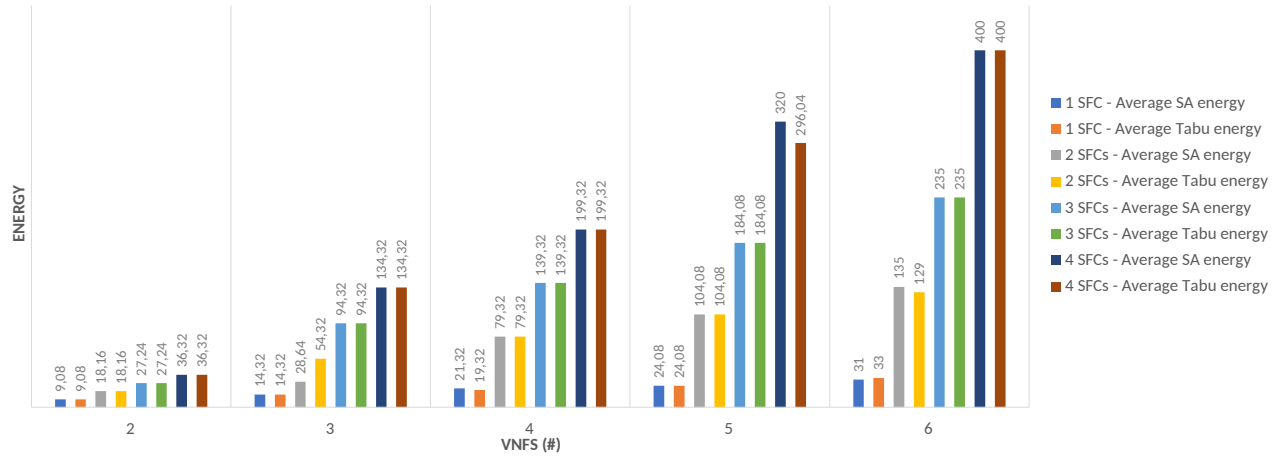


Fig. 5. Tabu Search and SA lowest energy values depending on the number of SFCs and VNFs per chain with a topology of 20 nodes.

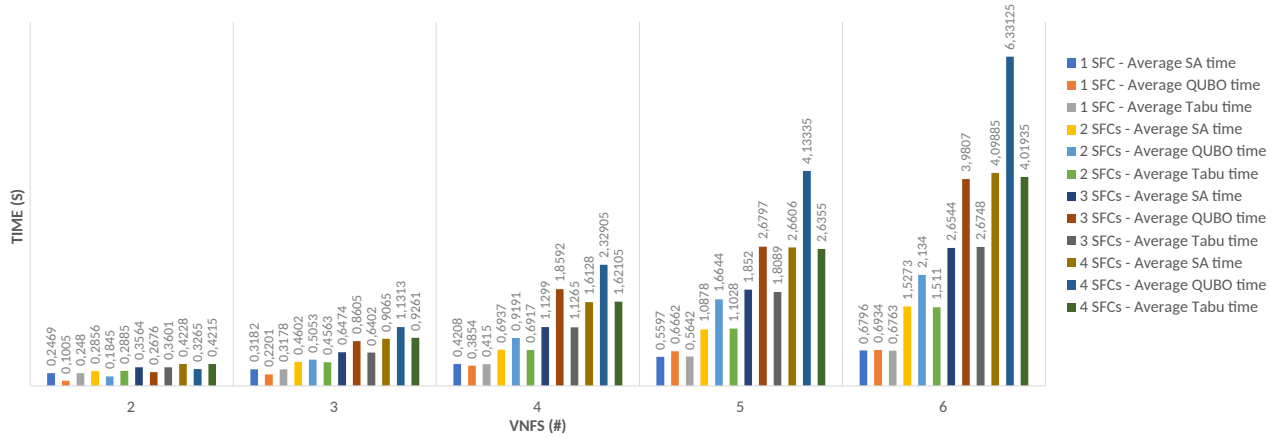


Fig. 6. Tabu Search and SA total time depending on the number of SFCs and VNFs per chain with a topology of 20 nodes.

QUBO problem is related to the constraints on node and link resources. This observation is important as it indicates that those constraints have a significant impact on the number of connections between QUBO variables and therefore increase the complexity of embedding it on the QPU resulting in a higher qubit requirement.

The full QUBO formulation is evaluated in Table IV. It can be seen how enabling the constraints related to the resources causes a significant increase in the number of both QUBO variables and qubits required. The increase in the number of QUBO variables is due to the need for slack variables to represent the inequality constraints on the resources. The number of qubits required for each QUBO variable increases drastically from the 2.87 of the previous experiment to 13.46. This is likely mostly due to the much higher connectivity of the QUBO problem (see Figure 7) which makes much more difficult to ensure all the needed physical connections are created on the QPU resulting in many more qubit chains to represent every single QUBO variable. In terms of the quality of the solution, QA is less effective compared to the classical solvers. Increasing the number of reads from $10^2$ to $10^4$ provides a limited improvement at the cost of substantially higher QPU solver time. It should be noted that the QPU solution with $10^2$ reads and energy of 204 violates the allocation constraint. The reason for the different effectiveness of QA in this case is likely due to the much higher connectivity of the QUBO problem coupled with the longer qubit chains and the need for several slack variables to represent the inequality constraints. The negative impact of long qubit chains for highly connected problems is a known issue. On one side the presence of chains makes the QA process more *rigid* because changing the state of a single qubit is energetically easier than changing the state of a dozen connected ones simultaneously. Furthermore, if a chain *breaks*, i.e., the qubits have different values, the resulting solution is likely suboptimal or may even violate constraints. In this respect, a future research question may be

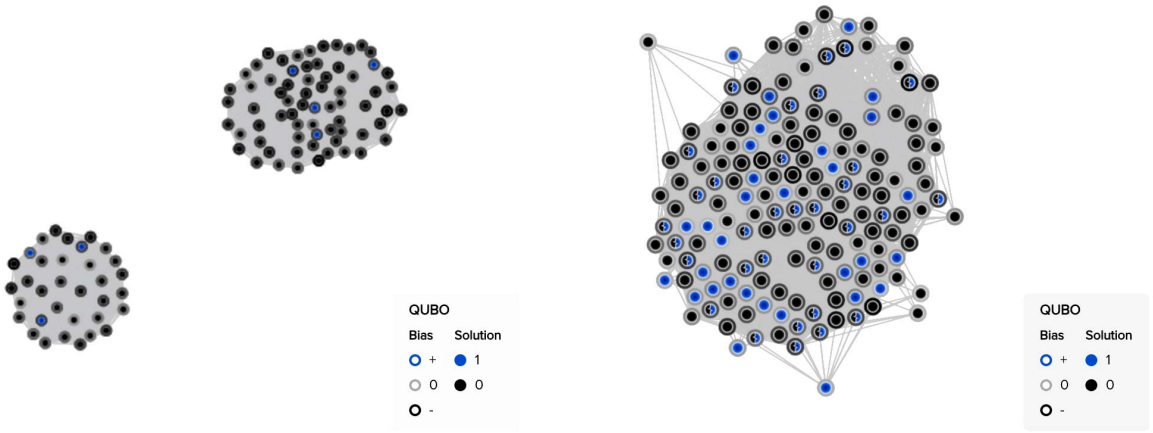


Fig. 7. Visualisation as a graph of the QUBO problem of Table IV. Each variable is a node and an edge exists between them if they appear in a quadratic term. On the left the experiment in only including allocation and continuity constraints; on the right the experiment with the full VNFEP QUBO formulation.

TABLE V
COMPARISON BETWEEN CLASSICAL SOLVERS AND QPU USING NET5 AND THREE SIMPLE SECAAS SERVICES.

| Problem | Solver | QUBO variables | Qubits | Solver time (s) | QUBO time (s) | Chain Strength | Lowest Energy | QA time (s) | Reads |
|---|---|---|---|---|---|---|---|---|---|
| Allocation and Continuity Constraints Only | SA | 72 | - | 0,1821 | 0,0021 | - | 0 | - | $10^2$ |
| | Tabu | 72 | - | 2,1947 | 0,0021 | - | 0 | - | $10^2$ |
| | QA | 72 | 250 | 0,0333 | 0,0021 | 50 | 0 | 50 | $10^2$ |
| Allocation and Continuity Constraints with Costs | SA | 72 | - | 0,1862 | 0.0047 | - | 27 | - | $10^2$ |
| | Tabu | 72 | - | 2,1319 | 0.0047 | - | 27 | - | $10^2$ |
| | QA | 72 | 252 | 0,0299 | 0.0047 | 50 | 27 | 50 | $10^2$ |
| Full VNFEP Formulation | SA | 128 | - | 0,3918 | 0.0355 | - | 27 | - | $10^2$ |
| | Tabu | 128 | - | 2,1600 | 0.0355 | - | 27 | - | $10^2$ |
| | QA | 128 | 1335 | 2,7550 | 0.0355 | 50 | 75 | 50 | $10^4$ |

improving the proposed VNFEP formulation aiming to reduce its connectivity and therefore improving its effectiveness.

The results of the analysis are consistent across the other networks. Table V shows the results obtained on the same experiments with the `net5` topology and allocating three `simple secaas` services. In this case as well all solvers are able to find a solution that satisfies the allocations and continuity constraints and, when the cost is included, all solvers find the same lowest energy solution. In this case QA is the fastest solver and is able to find the optimal solution with $10^2$ reads. When considering the full VNFEP formulation again the number of QUBO variables increases and so does the number of qubits required on average for each QUBO variable (from 3.47 to 10.42). This again results in reduced effectiveness for QA even with a high number of reads, $10^4$.

## VI. CONCLUSIONS

The emerging paradigm of Network Functions Virtualisation aims to transition from a network where functionalities are implemented via special-purpose hardware devices to a more flexible and scalable network where functionalities are implemented at a software level on general-purpose servers. This has created the need to develop efficient solutions to allocate these Virtual Network Functions on physical devices, accounting for several constraints related to the network structure as well as the resources available on the physical nodes, while simultaneously minimising costs.

This paper presents a novel formulation of the Network Functions Virtualisation Embedding Problem (VNFEP) as a QUBO problem that can be tackled with Quantum Annealers. The experimental analysis shows that when only continuity and allocation constraints are used, as well as costs, the QA performs at par with classical solvers, being able to find the optimal solution, although it is not consistently the fastest solver. However, including the resource constraints results in a highly connected QUBO problem and require slack variables to model inequalities. Due do this, representing the QUBO problem on the QPU becomes more difficult and requires a much higher number of qubits as well as longer qubit chains. This affects negatively the solution quality of QA compared with classical solvers.

Overall, the results indicate that tackling VNFEP using the QUBO formulation and the Quantum Annealer is possible and relevant for the specific domain of Network Functions

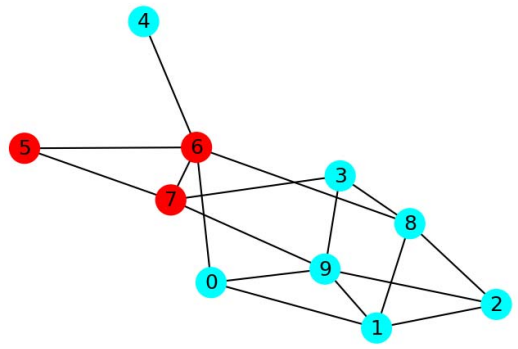


Fig. 8. Allocation results related to the experiment in Table IV. The VNFs of SFCs S0 and S1 are allocated on the red nodes inside the adopted network topology. On the bottom left, there is a complete description of the results of the embedding process and its energy. The two VNFs of S0 are allocated to nodes 6 and 7 and the three VNFs of S1 to nodes 5, 6, and again 5. The two virtual links of S1 are both embedded on the same physical link (5-6).

Virtualisation. Some challenges emerge due to the modelling of some constraints and the current technological limitations of available quantum annealers. Open research questions for future work are how to improve the VNFEP QUBO formulation in order to reduce its high connectivity as well as reduce the need for slack variables when the resource constraints are taken into account. Nevertheless, the results are promising and pave the way for further studies on the topic.

## REFERENCES

[1] M. A. Abdelaal, G. A. Ebrahim, and W. R. Anis, "Efficient Placement of Service Function Chains in Cloud Computing Environments," *Electronics*, vol. 10, no. 3, p. 323, Jan. 2021, number: 3 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2079-9292/10/3/323

[2] W. Xuan, Z. Zhao, L. Fan, and Z. Han, "Minimizing Delay in Network Function Visualization with Quantum Computing," *arXiv:2106.10707 [cs, eess, math]*, Jun. 2021, arXiv: 2106.10707. [Online]. Available: http://arxiv.org/abs/2106.10707

[3] B. Yi, X. Wang, and M. Huang, "Optimised approach for VNF embedding in NFV," *IET Communications*, vol. 12, no. 20, pp. 2630–2638, 2018, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/iet-com.2018.5509. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-com.2018.5509

[4] M. Asgarian, G. Mirjalily, and Z.-Q. Luo, "Embedding Multicast Service Function Chains in NFV-Enabled Networks," *IEEE Communications Letters*, vol. 25, no. 4, pp. 1264–1268, Apr. 2021, conference Name: IEEE Communications Letters.

[5] H. Ushijima-Mwesigwa, C. F. A. Negre, and S. M. Mniszewski, "Graph partitioning using quantum annealing on the d-wave system," *In Proceedings of the Second International Workshop on Post Moores Era Supercomputing (PMES '17)*, vol. abs/1705.03082, 2017. [Online]. Available: http://arxiv.org/abs/1705.03082

[6] C. Bauckhage, N. Piatkowski, R. Sifa, D. Hecker, and S. Wrobel, "A QUBO formulation of the k-medoids problem," in *Proceedings of "Lernen, Wissen, Daten, Analysen"*, ser. CEUR Workshop Proceedings, vol. 2454, 2019, pp. 54–63. [Online]. Available: http://ceur-ws.org/Vol-2454/paper_39.pdf

[7] D. Willsch, M. Willsch, H. D. Raedt, and K. Michielsen, "Support vector machines on the d-wave quantum annealer," *Comput. Phys. Commun.*, vol. 248, p. 107006, 2020. [Online]. Available: https://doi.org/10.1016/j.cpc.2019.107006

[8] S. H. Adachi and M. P. Henderson, "Application of quantum annealing to training of deep neural networks," *CoRR*, vol. abs/1510.06356, 2015. [Online]. Available: http://arxiv.org/abs/1510.06356

[9] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, "Quantum boltzmann machine," *Physical Review X*, vol. 8, no. 2, p. 021050, 2018.

[10] M. Ferrari Dacrema, F. Moroni, R. Nembrini, N. Ferro, G. Faggioli, and P. Cremonesi, "Towards feature selection for ranking and classification exploiting quantum annealers," in *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, E. Amigó, P. Castells, J. Gonzalo, B. Carterette, J. S. Culpepper, and G. Kazai, Eds. ACM, 2022, pp. 2814–2824. [Online]. Available: https://doi.org/10.1145/3477495.3531755

[11] R. Nembrini, M. Ferrari Dacrema, and P. Cremonesi, "Feature selection for recommender systems with quantum computing," *Entropy*, vol. 23, no. 8, p. 970, 2021. [Online]. Available: https://doi.org/10.3390/e23080970

[12] C. Carugno, M. Ferrari Dacrema, and P. Cremonesi, "Evaluating the job shop scheduling problem on a d-wave quantum annealer," *Nature Scientific Reports*, vol. 12, no. 1, p. 6539, Apr 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-10169-0

[13] Etsi gs nfv specifications. [Online]. Available: https://www.etsi.org/committee/nfv

[14] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[15] F. Glover, G. Kochenberger, and Y. Du, "Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models," *4OR*, vol. 17, Dec. 2019.

[16] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi, "Optimization by simmulated annealing," *Sci.*, vol. 220, no. 4598, pp. 671–680, 1983.

[17] G. Palubeckis, "Multistart tabu search strategies for the unconstrained binary quadratic optimization problem," *Ann. Oper. Res.*, vol. 131, no. 1-4, pp. 259–282, 2004. [Online]. Available: https://doi.org/10.1023/B:ANOR.0000039522.58036.68

[18] V. Choi, "Minor-embedding in adiabatic quantum computation: I. the parameter setting problem," *Quantum Inf. Process.*, vol. 7, no. 5, pp. 193–209, 2008. [Online]. Available: https://doi.org/10.1007/s11128-008-0082-9