Engineering Grover Adaptive Search: Exploring the Degrees of Freedom for Efficient QUBO Solving

Engineering Grover Adaptive Search: Exploring the Degrees of Freedom for Efficient QUBO Solving

Luigi Giuffrida, Deborah Volpe, *Graduate Student Member, IEEE*, Giovanni Amedeo Cirillo, *Graduate Student Member, IEEE*, Maurizio Zamboni, and Giovanna Turvani

***Abstract*—Quantum computers have the potential to solve Quadratic Unconstrained Binary Optimization (QUBO) problems with lower computational complexity than classical ones. Considering the current limitations of quantum hardware, the joint use of classical and quantum paradigms could exploit both advantages. Quantum routines can make some complex tasks for classical computers feasible. For example, in the Grover Adaptive Search (GAS) procedure, the problem cost function is classically shifted iteratively, whenever a negative value is found through the quantum Grover Search (GS) algorithm, until the minimum is achieved. This quantum-classical approach is characterized by many degrees of freedom, e.g. the number of GS iterations in each call and the stop condition of the algorithm, which should be appropriately tuned for an effective and fast convergence to the optimal solution. The availability of software routines could permit the best management of the GAS parameters.**

This work proposes new mechanisms for GAS parameters management and compares them with the existing ones, like one available in the Qiskit framework. The proposed mechanisms can automatically arrange the parameters according to the algorithm evolution and their previous experience, thus ensuring a more frequent and faster achievement of the optimal solution.

Even though these strategies can be further improved, the results are encouraging. The analysis is done to identify the best policy for different problems. It lays the foundation for designing an automatic toolchain for QUBO solving, which can obtain the best possible implementation of the GAS algorithm for each submitted problem.

***Index Terms*—Grover Search, Grover Adaptive Search, Hybrid Quantum-Classical Algorithms, Optimization Problems, Quadratic Unconstrained Binary Optimization, Cost Function, Quantum Dictionary.**

I. Introduction

COMBINATORIAL optimization (**CO**) problems aim to find an input configuration that minimizes a **cost function**. They are relevant in many real-world applications, such as resource allocation in industrial environments.

The optimal solution can always be found with a **brute-force approach**, i.e. by testing all possible input variables combinations, but the time required increases exponentially with them, thus making the scalability of this mechanism unfeasible. Deterministic exploration of the solutions space, e.g. based on the gradient computation, can also be exploited. However, this is unsuitable for some optimization problems, such as multimodal ones and can require a significant amount of time to achieve convergence. Because of their limitations and the unfeasibility of brute-force exploration, heuristic approaches are commonly employed to find optimal or sub-optimal solutions for large-scale problems. Although many new classic approaches have been proposed in recent years, they are not always satisfactory in terms of execution time or accuracy. Therefore, the exploitation of quantum computers was proposed to obtain a speed-up by exploiting its intrinsic parallel computational capabilities due to superposition and entanglement principles [1], [2].

The most feasible formulation for solving CO problems with quantum computers is the **Quadratic Unconstrained Binary Optimization** (QUBO) one, presented in Section II-A. Quantum hardware fabrication is in the middle of the so-called Noisy Intermediate-Scale Quantum (NISQ) era, which is characterized by the availability of general-purpose quantum computers, based on the so-called circuit model paradigm, with a limited number of qubits subjected to non-ideality phenomena, thus limiting the scalability of the executable algorithms. However, the Yole [3] report on quantum technology shows how the circuit-model quantum computers are expected to scale rapidly in the following years. For this reason, even though hardware fabrication has not yet achieved complete maturity, the evaluation of circuit-model-based QUBO solvers can already be done.

A further possibility is to use jointly classical and quantum computers to make the best of both paradigms. In consolidated hybrid quantum-classical algorithms, quantum routines are exploited for accelerating some specific tasks critical or unfeasible for classical computers. A quantum processing unit, in this context, can be seen as a hardware accelerator, particularly effective in solving a specific task. One of the most relevant algorithms in this context is the **Grover Adaptive Search** (**GAS**) [4], whose analysis and improvement are the target of this work. It exploits the **Grover Search** (GS) algorithm for finding a negative value of the optimization problem cost function, which is mapped onto a quantum circuit. Then the function is classically shifted by an amount equal to the last negative sample measured; this sequence is repeated until the minimum is found.

This work is going to **propose new approaches for managing the GAS degrees of freedom** — i.e. the number of GS rotations in each call and the mechanism for stopping the algorithm, able to automatically change the parameters

The authors are with the Department of Electronics and Telecommunications of Politecnico di Torino, Torino, 10129, Italy (e-mail: luigi.giuffrida@studenti.polito.it; deborah.volpe@polito.it; giovanni_cirillo@polito.it;
maurizio.zamboni@polito.it; giovanna.turvani@polito.it.)

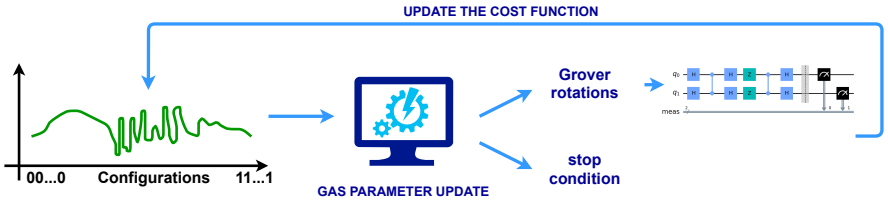


Fig. 1: Grover Adaptive search closed-loop. At each classical iteration, the software routine maps the updated cost function onto the quantum circuit, generates the Grover circuit with the optimal length, i.e. the number of iterations, and checks the stop condition according to a strategy depending on the characteristics of the problem cost function.

according to the algorithm evolution and previous experience — and to **compare them with the existing ones**, to prove their effectiveness and efficiency in solving different types of QUBO problems. The purpose is to identify the best strategies combination for each considered problem type, which is fundamental for the algorithm effectiveness, strictly depending on the capability to well-identify them for the case of interest. The performed analysis is particularly of interest to the design automation community because it lays the fundamentals for implementing in the future an **automatic toolchain for QUBO solving with quantum computers**, which — as shown in Figure 1 — can design and realize the GAS algorithm, with its quantum (circuit design) and classical (stop condition) constituting parts, in the most effective way for every considered problem. **This work is perspective**, considering what has already been said about hardware limitations.

The article is organized as follows. Section II reports theoretical foundations; in particular, the QUBO formulation and the GAS behaviour are explained. Section III reports the proposed strategies for degrees of freedom management. In Section IV, simulation and results discussion are presented. Finally, in Section V, conclusions are drawn, and future perspectives are illustrated.

II. Theoretical Foundations

A. QUBO formalism

Quadratic Unconstrained Binary Optimization (QUBO) is a mathematical formulation able to represent an exceptional variety of CO problems [5]. **Quadratic** refers to the highest power applied on the involved variables, which can assume only 0 and 1 values (thus **Binary**). **Unconstrained** means that no constraints are applied, and **Optimization** is related to the fact that this model is used to minimize the obtained objective function, which is written as:

$$\text{Obj}(c, a_i, b_{ij}, x_i) = c + \sum_i x_i \cdot a_i + \sum_{i<j} b_{ij} \cdot x_i x_j \,, \qquad (1)$$

where $x_i \in [0, 1]$ is a binary variable, $x_i x_j$ is a coupler that allows two variables to influence each other, a_i is a weight or bias associated with a single variable and b_{ij} is a strength which controls the influence of variables i and j.

Despite what the name would suggest, problems with constrained solution space can also be represented as QUBO by introducing quadratic penalty to the objective functions:

$$\text{minimize} \quad y = f(\mathbf{x}) + \lambda g(\mathbf{x}) \,, \qquad (2)$$

where λ is a positive penalty weight assigned to the constraint quadratic function $g(\mathbf{x})$. In this way, the constraint is evaluated during the standard execution of the optimizer, assigning a higher value to the solution $\mathbf{x}$ that does not satisfy the constraint. Therefore, the choice of λ is critical: a too high value makes the cost function too flat for evaluating the effective quality of a feasible solution than the others, while a too low value makes the constraints negligible, thus allowing the acceptance of unfeasible solutions.

QUBO formulation can be assisted by Python libraries, such as **qubovert** [6], which is used for benchmark problem formulation in this paper. It is characterized by routines for automatic insertion of some relevant constraints in the problem function and the possibility to interface the defined QUBO problems with different QUBO solvers.

1) Relevant Examples: Some relevant optimization problems representable with QUBO formulation, employed as benchmarks in this article, are described in the following.

The goal of **max-cut** problems [5] (Figure 2a) is to partition a graph into two complementary subsets S and $\overline{S}$, maximizing the sum of weights over all the edges across the two vertices subsets. Its QUBO formulation involves a binary variable for each node, whose value is 1 or 0, depending on the subset to which the node belongs. The quantity $x_j + x_i - 2x_j x_i$ identifies whether the edge (i, j) is in the cut. In particular, it is equal to 1 when one among x_i and x_j equals 1, so when edge (i, j) is effectively in the cut. Summing the contributions of each edge, the following QUBO formulation can be obtained:

$$\text{Maximize} \quad y = \sum_{(i,j)\in E} w_{i,j} \cdot (x_j + x_i - 2 \cdot x_j \cdot x_i) \,, \qquad (3)$$

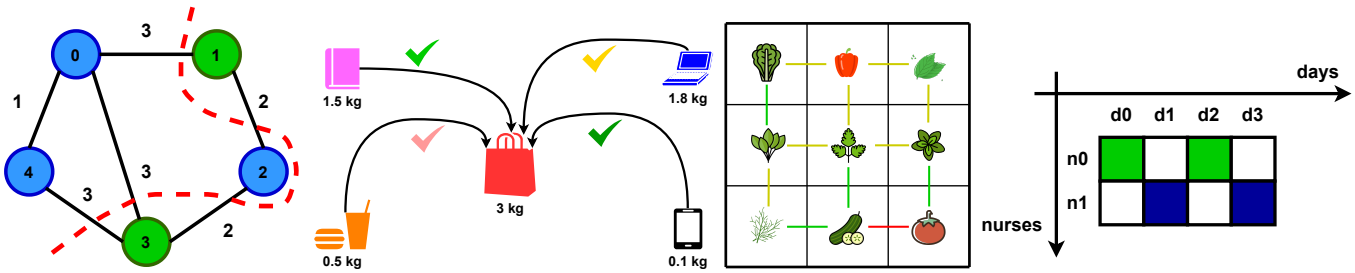
where $w_{i,j}$ is the weight of the edge that connects the i-th and the j-th node. Max-cut problems are characterized by symmetric energy profiles; in fact, a solution and its complement (e.g. [0,1,1,0,1] and [1,0,0,1,0]) have the same energy because the obtained two sub-sets are interchangeable.

In the case of **knapsack** [7] problem (Figure 2b), the target is defining, for a set of objects x_i characterized by weight w_i, the best sub-set to be put into a *bag*, ensuring that the total weight does not exceed a threshold W:

$$0 < \sum_{i=1}^{\dim(X)} w_i x_i \leq W \,, \qquad (4)$$

and maximizing the number of *favorite* objects (the preference of each object is expressed through a weight p_i).

The reported inequality constraint can be represented in



(a) Five-node max-cut. (b) Knapsack. (c) Garden Optimization (d) Nurse scheduling

Fig. 2: Examples of QUBO problems.

QUBO formulation by using auxiliary variables whose number depends on W, as explained in [5].
The cost function can be finally written as:

$$f_{\text{knapsack}}(\mathbf{x}) = f_{\text{inequality}}(\mathbf{x}) - \sum_i p_i x_i \,. \tag{5}$$

Another important CO problem family is that of **well-positioning problems**, such as the **garden optimization** [8], whose target is to find an optimal placement of n plants in n pots (Figure 2c). The associated QUBO problem involves n^2 variables x_{ij} for each plant-pot pair, assuming value 1 if the j plant is in the pot i. A valid placement has to satisfy some requirements. First of all, each pot has to be filled with exactly one plant:

$$\forall i: \quad \sum_{j=1}^{n} x_{ij} = 1 \,. \tag{6}$$

Then, all available plants must be placed in the garden:

$$\forall j: \quad \sum_{i=1}^{n} x_{ij} = 1 \,. \tag{7}$$

In the end, tall plants shall not shadow smaller ones:

$$\forall i,j: \quad (i \mod 2 - s_j)^2 x_{i,j} = 0, \tag{8}$$

where $s_j \in [0,1]$ is a binary flag assuming value 0(1) if the j^{th} plant is tall(small), forcing it in even(odd) rows.
The figure of merit for placement optimization is the affinity among plant species. Indeed, some species can be placed close to each other, while others cannot.
The final cost function can be written as:

$$\begin{aligned} f_{\text{garden}}(\mathbf{x}) = &- \sum_{i,i'=1}^{n} J_{ii'} \left(1 + \sum_{j,j'=1}^{n} x_{ij} C_{jj'} x_{i'j'}\right) + \\ &+ \lambda_1 \sum_{i=1}^{n} \left(1 - \sum_{j=1}^{n} x_{ij}\right)^2 + \lambda_2 \sum_{j=1}^{n} \left(1 - \sum_{i=1}^{n} x_{ij}\right)^2 + \\ &+ \lambda_3 \sum_{i=1}^{n} \sum_{j=1}^{n} (i \text{ rem } 2 - s_j)^2 x_{ij}, \end{aligned} \tag{9}$$

where $J_{ii'}$ and $C_{jj'}$ are the terms of the adjacent J and companions C matrices, respectively. $J_{ii'}$ is equal to 1 if pots i and i' are adjacent, while $C_{jj'}$ is equal to -1 when there is a friendly relationship among plants j and j' plants, and equal to 0 and 1 in case of neutral and antagonist relationships respectively.
Finally, the **timetabling problems**, like **nurse scheduling optimization** [9], are described. The target is to find the optimal schedule for nurses working in a hospital over a fixed timetable of shifts (Figure 2d). Given N nurses and D working days, the associated QUBO model involves $N \cdot D$ variables, one for each nurse-day pair, which equals 1 if the n^{th} nurse works on the d^{th} day. Three constraints characterize this problem. The first one is called *hard shift constraint* and requires that the schedule has to assure in each day d a nurse effort $\sum_{n=1}^{N} E(n) x_{n,d}$ sufficient to satisfy the associated workload $W(d)$:

$$\forall d: \quad \sum_{n=1}^{N} E(n) x_{n,d} = W(d) \,. \tag{10}$$

The second one is called *hard nurse constraint*, which ensures that no nurse works for two consecutive days. A positive correlation constant a is used to penalize the schedule in which the n^{th} nurse works on two consecutive days:

$$f_{\text{nurse}}(\mathbf{x}) = \sum_{n=1}^{N} \sum_{d=1}^{D-1} a \cdot x_{n,d} \cdot x_{n,d+1} \,. \tag{11}$$

The last one, which is called the *soft nurse constraint*, assures that all nurses should work approximately the same number of days $F = D/N$:

$$\forall n: \quad \sum_{d=1}^{D} x_{n,d} = F \,. \tag{12}$$

The final cost function can be written as:

$$\begin{aligned} f_{\text{nurse}}(\mathbf{x}) = \sum_{n=1}^{N} \sum_{d=1}^{D-1} a x_{n,d} x_{n,d+1} + \lambda_1 \sum_{n=1}^{N} \left(\sum_{d=1}^{D} x_{n,d} - F\right)^2 + \\ + \lambda_2 \sum_{d=1}^{D} \left(\sum_{n=1}^{N} E(n) x_{n,d} - W(d)\right)^2 . \end{aligned} \tag{13}$$

All the previously described CO problems are employed as benchmarks for the proposed Grover-based QUBO solvers, and the results for these are reported in Section IV.

B. Grover Adaptive Search

A CO problem, written as a cost function $f(x)$:

$$\text{minimize} \quad f(x), \tag{14}$$

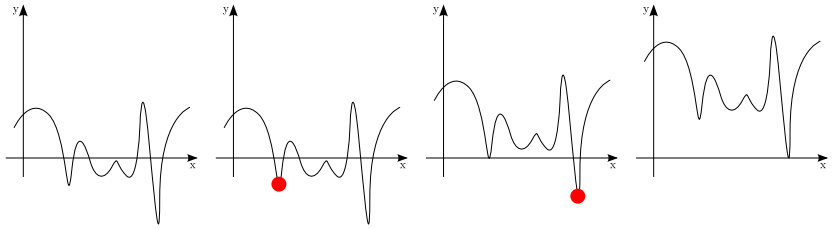


Fig. 3: Example of the iterations of the described algorithm: the cost function is shifted twice and, after the second shift, its lowest value is equal to 0.

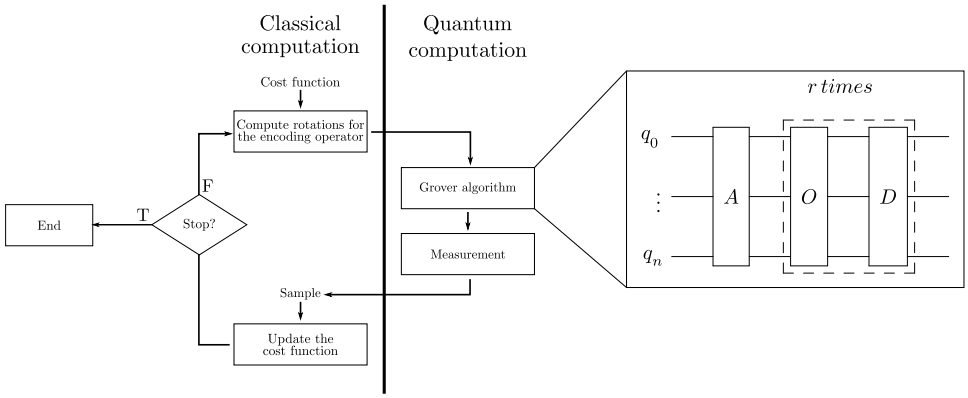


Fig. 4: Flowchart of the GAS algorithm, with focus on GS circuit.

can be solved with the following sequential approximation method:

1) Set to zero the iteration index i and the last sampled value variable y_i.
2) Shift the cost function $f(x)$ of y_i vertically.
3) Increase i.
4) Randomly sample a value y_i from the image of $f(x)$ such that $y_i < 0$.
5) Assume y_i as a new optimal value since a negative value of the shifted cost function is consistently lower than the previously imposed offset y_{i-1}.
6) Repeat from 2 until no more negative values are sampled. In this way, the combination of the variables with $f(x) = 0$ is the solution to the optimization problem.

A meaningful example of this approach is reported in Figure 3, where it is possible to observe how the minimum can be found by repeating the sample-and-shift procedure. In order to properly exploit the described procedure, an efficient mechanism for sampling negative values is required. **Grover Search** (**GS**) routine, which is particularly efficient compared to its classical counterpart in finding a particular item in an unordered set (described in detail in Appendix A), represents the best choice for this task. If this quantum algorithm is exploited, the global optimization problem solver is called **Grover Adaptive Search** (**GAS**) [10], and its complete flowchart is reported in Figure 4.

The cost function can be described properly for the GS routine by encoding it onto a quantum state in terms of the key-value pairs of a quantum dictionary, which is the quantum counterpart of a classical dictionary and is described in detail in Appendix B. In this context, it is sufficient to say that keys and values correspond to the cost function domain (i.e. combinations of binary variables $\mathbf{x}$) and image ($f(\mathbf{x})$, described according to the binary two's complement representation for signed integers), respectively.

Another critical point of the presented procedure is determining if there are still negative values to be sampled or not. In the GAS case, a characteristic feature of GS can be exploited for this purpose: when no item meets the conditions of the GS, any possible configuration can be sampled according to a uniform probability distribution. This occurs when only non-negative function values are available, so GAS can be stopped when this condition is achieved since the optimal value has already been obtained. However, this is not the only situation in which a positive value can be obtained. Indeed, it can occur when a wrong number of Grover rotations is chosen. Consequently, a good possibility is to count the number of consecutive measured positive samples and to stop the algorithm when this overcomes a certain threshold t, after which the condition of optimal solution achievement can be reasonably trusted. The optimization of t is discussed in Section II-C.

C. GAS degrees of freedom

Given the number of iterations of the Grover algorithm $r \in \mathbb{N}$, its optimal value for measuring one desired item depends on the fraction of labelled states over the whole search domain. The probabilities of measuring labelled or non-labelled states depend on the chosen r:

$$\gamma\left(\{x\}\right) = \begin{cases} \frac{g_r(p)}{l} & \text{if } x \in M \\ \frac{1-g_r(p)}{N-l} & \text{if } x \in S \setminus M \end{cases}, \tag{15}$$

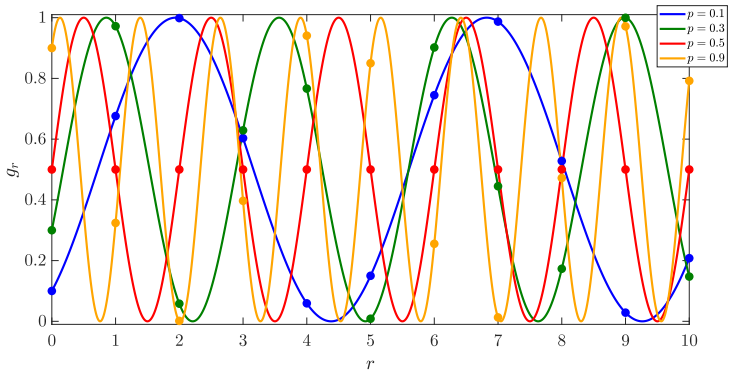


Fig. 5: Dependence of g_r on r, for four different p values. Since r must be a natural number, dots mark the effective g_r.

where l is the number of labelled states, N is the number of elements in the searching domain, $p = \frac{l}{N}$ is the fraction of labelled states in the domain, M is the set of target states, and S is the total searching domain. The function $g_r(p)$ is given by:

$$g_r(p) = \sin^2\left[(2r+1)\arcsin\sqrt{p}\right] . \tag{16}$$

Figure 5 shows the relationship between g_r and r for four different p values. It is possible to observe how the period of g_r decreases by increasing p, even if this does not necessarily imply that a solution can be found with the Grover algorithm in fewer iterations. Indeed, only integer values of r, marked by dots, can be effectively employed for GS. As it is possible to notice, the GS is more effective when the searched items are less than half of the possible solutions. An unfortunate case is when the targets are exactly half of the solution space because g_r is constantly equal to 0.5, giving each item in the solution space the same probability of being sampled.

In the GAS context, M contains negative cost function values at each iteration. Its dimension, thus the percentage of target state p, cannot be known, and they change whenever the cost function is translated. For this reason, the choice of r is critical, and some strategies to choose it are required.

From Equation 15, it is also possible to notice that a positive sample can be obtained in two cases: no negative values remain, or the number of Grover rotations r is not well-selected. Therefore, to correctly stop the Grover algorithm, it is necessary to develop an effective mechanism to distinguish these two situations. As mentioned in Section II-B, the most common solution is to count the number of consecutive positive samples until a certain threshold t is reached.

For all the mentioned reasons, the number of Grover rotations at each algorithm call and the stop condition mechanism are the most crucial degrees of freedom of the GAS algorithm for its effective execution.

III. Explored Techniques for Managing GAS Degrees of Freedom

As described in Section II-C, optimizing the algorithm degrees of freedom is crucial. In particular, the number of Grover rotations r has to be estimated at each GS call because the percentage of cost function negative values changes after each translation. Moreover, the stop condition threshold t has to be chosen to find a compromise between short total execution time and satisfactory GAS success probability.

This section presents existing and new mechanisms for managing the degrees of freedom. Furthermore, to permit a consistent results comparison, corresponding software methods have been added to the **GroverOptimizer class in Qiskit** [11] (codes are available in https://github.com/DeborahVolpe/Grover-Adaptive-Search.git).

A. Grover iteration

Regarding the number of Grover rotations, the target is to determine the lowest value, ensuring a satisfactory probability of measuring the existing negative values.

Three strategies are presented: **the first two are already available in state-of-the-art, while the last one is presented here for the first time**. They are detailed in the following:

- **Random** (**RND**) approach [12], in which r is randomly sampled from an interval of possible values. This is initialized to $[0, 1]$, whenever the cost function is shifted (i.e. a negative value is measured), and is increased at every positive value sampled. The main idea behind this approach is to randomly explore $g_r(p)$ (Equation 16) until a value of r, permitting to obtain a negative value, is found. This is the mechanism already present in the Qiskit (**QSKT**) implementation, where the upper bound of the interval is saturated.
- **Fixed pattern** (**PAT**) proposed in [13], where r at each GS call is read by a list of pre-computed values, which are obtained by making some assumptions on the distribution of values in the image of the cost functions. However, this seems ineffective when there is a great deal of repetition in objective values, as shown in [14].
- **Linear** (**LIN**) approach, where r is set to 0 when a negative sample is obtained and is linearly increased whenever a positive one is measured. This mechanism, like the random one, explores $g_r(p)$ (Equation 16) until a negative value is met.

Both random and linear strategies have been implemented with (**noSat**) and without the following saturation (**sat**) mechanism:

$$m_{\text{new}} = \min\left\{m_{\text{old}} + 1,\, 2^{n_{\text{key}}/2}\right\}, \tag{17}$$

where n_{key} is the number of binary variables of the problem of interest and m_{new} and m_{old} depend on the strategy. In the linear one, m_{new} and m_{old} are the new and previous r, while in the random strategy m_{new} and m_{old} are the new and previous upper limits of the interval in which r is chosen randomly.

B. Stop policy

As previously explained, the common strategy for stopping GAS is to count the number of successive positive numbers until a threshold t is reached. However, choosing the best t is crucial for a compromise between GAS success probability and execution time. In the state-of-the-art, **this value is usually fixed for the whole execution**, but it does not consider the evolution of the cost function during the GAS execution. Sampling a positive value at the beginning of the algorithm could imply a bad choice of r rather than a total presence of positive values, while at the end of GAS, the situation is dual. For these reasons, **dynamic mechanisms for varying** t are proposed in this work:

- **Linear** (**LIN**), where the t is linearly decreased, starting from a reasonably high value $t_{\max}$, whenever a negative value of the cost function is measured. The t update equation is:

$$t_{\text{new}} = \left\lceil -\frac{t_{\max}}{t_{\min}} \frac{n_S}{v} + t_{\max} \right\rceil , \tag{18}$$

  where $t_{\min}$ and $t_{\max}$ are the minimum and maximum values of t, n_S is a counter of the executed classical iterations (i.e. of negative values sampled), and v is a control parameter of the linear scaling slope.
- **Logarithmic** (**LOG**), analogous to the previous one and employing a logarithmic decrease of t according to:

$$t_{\text{new}} = \left\lceil \frac{\log_{10}(1.1)(t_{\max} - t_{\min})}{\log_{10}(1.1 + n_S/v)} + (t_{\min} - 1) \right\rceil , \tag{19}$$

- **Adaptive** (**ADPT**), which permits to modify t according to the previous experience. In particular, when a negative value is measured after many consecutive positives close to t (currently $> \frac{4}{5}t$, but tunable), a too low threshold could be expected, so t must be increased by a fraction of t (currently 20%); on the other hand, a negative sample after few positives (currently $< \frac{1}{5}t$, but tunable) could imply a too high value for t, which can be reduced (currently 20%) to speed-up the execution.

In all cases, to avoid too low or negative values of t, this is saturated to the lowest expected value $t_{\min}$.

IV. Results

In this section, the most significant results — more precisely the most complex problem solved for each family — are reported and commented to understand the main observed trends in solving different QUBO problems. All the obtained results not reported in the following are available in the supplementary information file, with the same format (in terms of plots and tables) of those shown in the following.

All tests are performed by using for the quantum part the **QasmSimulator** available in Qiskit, which runs locally, and by using as execution platform a single-process Intel(R) Xeon(R) Gold 6134 CPU @ 3.20GHz opta-core, Model 85 [15] with a memory of 10296102+ KiB. The size of considered problems is limited by the number of simulable qubits on the available platform.

Table I reports, for each combination of strategies for managing GAS parameters, the statistical results obtained by repeating the test ten or one hundred times, according to the complexity of the benchmark problem. In particular, the **probability of success** (**POS**) — i.e. the probability of obtaining the optimal solution — **the average number of total Grover rotations** per GAS execution $R = \sum_{\textbf{GS calls}} r$, and the **average number of classical iterations** (C) are reported, to compare the results obtained with the different mechanisms both in terms of the effective capability of reaching the minimum of the cost function and of time required. The last two metrics give an idea of the computational complexity of a GAS-based QUBO solver. R reports the average value of oracle and diffusion operators in a complete GAS execution, thus permitting us to understand the total amount of quantum computation required for obtaining a solution. At the same time, C corresponds to the total number of the complete GAS iterations, each constituted by a GS quantum circuit execution and the following classical post-processing mechanism (Figure 4). For example, many total Grover rotations and classical iterations imply a long execution time. Qiskit implementation results are reported on top because they are considered the reference for comparisons. In order to more easily understand which is the best compromise between success probability and execution time, the obtained results are also graphically reported in Figures 6a, 6b, 6c and 6d. These report, for each strategy, the **cumulative distributions** obtained by executing GAS multiple times, **divided by** R. In order to facilitate the understanding of plots, two rules must be considered. According to the first one, the probability of obtaining the optimal value (or a value close to it) with a specific strategy is higher when its corresponding cumulative distribution is more concentrated on the left of the plot, where the lowest values of the cost function are located. According to the second one, the more the distribution is shifted up, lower is the total execution time, because the multiplicative factor $1\frac{1}{R}$ is higher. In other words, the cumulative distribution is a useful method for detecting the most accurate strategies and the scaling factor $\frac{1}{R}$ permits the distinction of two eventual strategies with analogous distributions in terms of the expected execution time for obtaining a solution. Moreover, to make more accessible the comparison with Qiskit, the region of the plot below its cumulative is coloured with the same colour.

From the reported results, it is possible to notice how the best strategies combination strictly depends on the considered type of problem, thus from the shape of its cost function. Indeed, pattern strategy seems to be not particularly effective in the case of the objective function with many value repetitions, like max-cut, but works quite well with the other tested problems, whose cost function shape is heterogeneous, coherently with the results reported in [14].

Linear and random strategies for choosing r provide a high success probability in all considered problems, while it could

TABLE I: Statistical results concerning the effectiveness of each strategy combination. In particular, the **probability of success** (**POS**), the **average number of total Grover rotations** (**R**) and **average number of classical iterations** (**C**) in a GAS execution are reported. The reported data are obtained by solving, with each strategies combinations, ten times an eight-node max-cut ($w_{i,j}$ = rand(range(0, 2))) and a four-object knapsack (seven-variable, considering those auxiliary, p_i = rand(range(1,3)), w_i = rand(range(1,3)) and W = rand(range(3,5))) problems and one hundred times a four-variable garden ($\lambda_1 = 10$, $\lambda_2 = 10$, $\lambda_3 = 5$ and $C_{jj'}$ taken from from [8]) and a four-variable nurse problems (N = 2, D = 2, $W(d) = 1$, $E(n) = 1$, $a = 3.5$, $F = 1$, $\lambda_1 = 0.3$ and $\lambda_2 = 1.3$).

Adopted strategy		**Max-cut with 8 variables**			**Knapsack with 7 variables**			**Garden with 4 variables**			**Nurse with 4 variables**		
r	**t**	**POS**	**R**	**C**	**POS**	**R**	**C**	**POS**	**R**	**C**	**POS**	**R**	**C**
QSKT	–	1.00	35.00	22.80	0.00	78.50	57.60	1.00	10.31	12.65	0.65	7.22	12.39
RND sat	**ADPT**	0.70	12.30	13.30	0.30	14.00	16.40	0.93	4.53	8.34	0.69	3.00	7.35
	LIN	0.40	4.00	8.50	**0.30**	**4.10**	**8.20**	0.82	3.26	7.02	**0.71**	**2.31**	**6.70**
	LOG	**1.00**	**16.40**	**15.90**	0.00	36.90	32.10	**0.96**	**8.50**	**11.05**	0.68	5.63	10.47
RND noSat	**FIX**	1.00	25.00	19.40	0.20	69.50	51.70	1.00	15.94	13.11	0.59	9.10	13.73
	ADPT	0.80	11.20	12.70	**0.50**	**11.20**	**14.00**	0.89	4.57	8.49	0.67	3.46	7.67
	LIN	0.20	4.30	8.50	0.10	9.20	11.00	0.89	3.12	7.08	**0.74**	**2.16**	**6.65**
	LOG	**1.00**	**19.00**	**16.80**	0.40	42.40	36.10	**0.98**	**10.53**	**11.08**	0.7	4.93	9.87
LIN sat	**FIX**	1.00	52.40	17.60	0.20	80.60	47.30	0.99	23.69	12.27	0.48	22.18	16.36
	ADPT	**0.90**	**13.20**	**11.70**	0.20	19.40	15.90	**0.98**	**5.77**	**7.52**	0.62	6.28	8.22
	LIN	0.40	7.00	8.40	**0.30**	**7.60**	**9.30**	0.93	5.73	7.01	**0.67**	**4.45**	**7.29**
	LOG	1.00	33.70	15.90	0.00	56.40	34.10	1.00	17.65	10.70	0.62	11.78	12.28
LIN noSat	**FIX**	1.00	49.70	15.60	0.20	80.20	46.50	1.00	31.00	12.63	0.50	23.50	17.45
	ADPT	**0.80**	**6.10**	**9.20**	0.10	13.80	14.40	**0.99**	**7.04**	**7.67**	0.71	4.73	7.72
	LIN	0.80	7.30	9.00	**0.20**	**8.50**	**9.20**	0.97	6.30	7.10	**0.73**	**3.78**	**7.15**
	LOG	1.00	32.30	16.10	0.10	69.30	42.10	1.00	19.38	10.80	0.60	12.38	12.03
PAT	**FIX**	**0.90**	**30.60**	**20.10**	**0.40**	**63.50**	**26.10**	0.99	8.07	12.61	0.66	15.13	26.10
	ADPT	0.60	8.60	12.70	0.10	10.70	14.50	**0.98**	**2.86**	**8.35**	0.60	1.77	14.50
	LIN	0.30	4.30	9.30	0.20	3.40	9.00	0.89	2.19	7.33	0.58	1.25	9.00
	LOG	0.70	11.70	14.50	0.30	55.00	24.30	0.99	5.57	11.12	**0.72**	**4.52**	**24.30**

become very high with the fixed threshold strategy.

The proposed mechanisms for dynamically modifying t are effective because they significantly reduce R without unduly affecting the success probability. Also, in this case, the best trend variation of t depends on the problem, but on average, the logarithmic decrease and the adaptive policy results are the most effective.

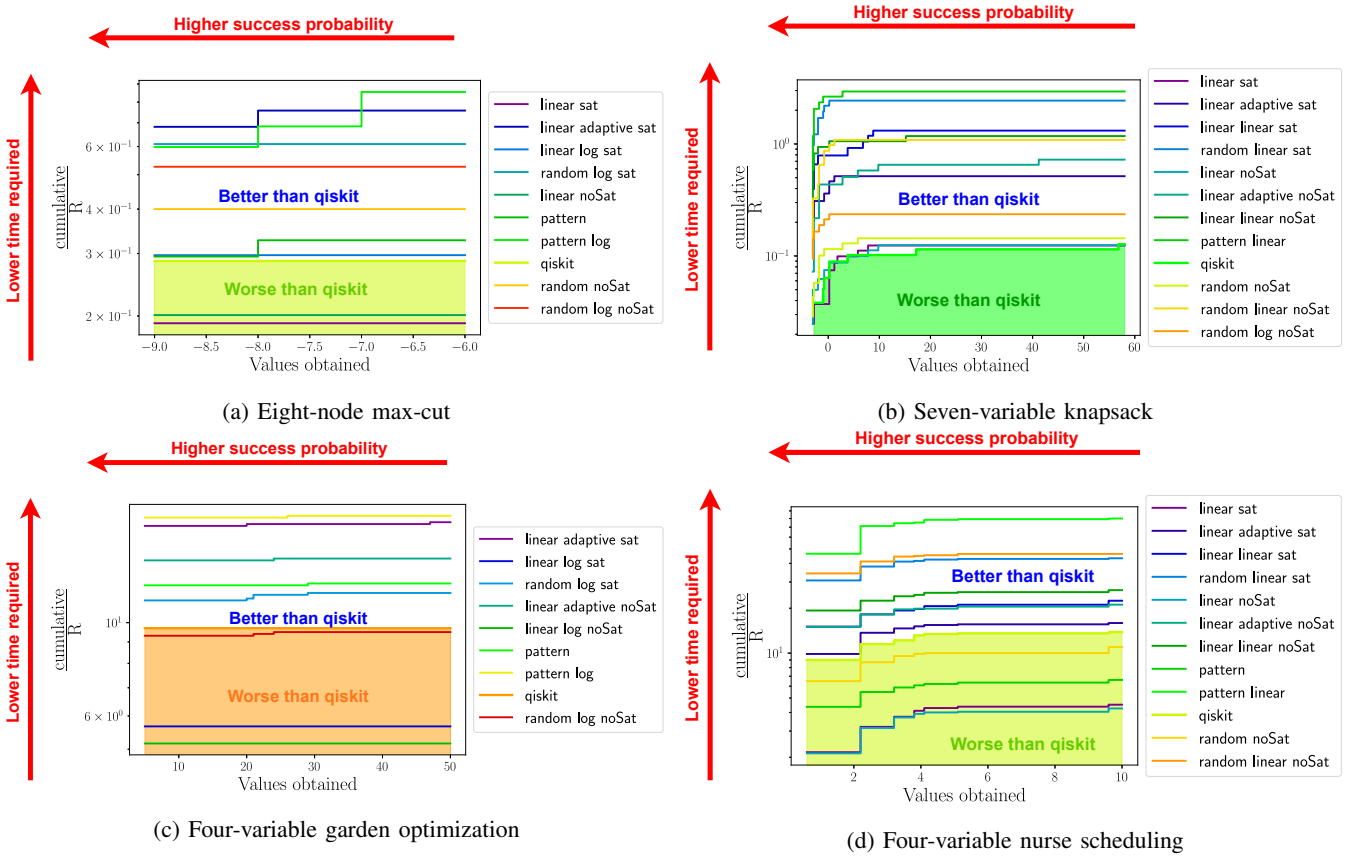
V. Conclusion

This work proposed new strategies for managing GAS degrees of freedom and compared them with those in state-of-the-art solving QUBO problems. The results show how the proposed mechanism for dynamically modifying t permits a significant reduction of R, i.e. the execution time, without excessively degrading the success probability. However, coherently with expectation, the best strategy combination is strongly correlated with the cost function of the problem. For example, the pattern strategy for updating r is particularly effective with nurse scheduling problems, characterized by few repetitions in the objective function, but ineffective with max-cut ones, where at least two repetitions of each value of the cost image are present. In any case, the obtained results can be considered reasonable and encouraging.

In order to further improve the GAS algorithm, other strategies for managing all its degrees of freedom should be explored in the future. For example, the stop condition mechanism could be improved by performing a final Grover search with a different oracle when the threshold t is met. This could label non-negative values when more than half of the solution space is of this type. In fact, as explained in Section II-C, if states to be labeled are more than half of the solution space, Grover search amplifies the probability of non-labeled states, which can be sampled with a high probability for low $r > 0$ (see in Figure 5 the trend of the dual curves associated with $p = \{0.1; 0.9\}$ for $0 \leq r \leq 2$). Another possibility, along the lines of the previous one, could be to employ an oracle labelling negative and null values. This could be employed when the absolute minimum, whose value is 0 at the end of GAS, has been achieved or is very close.

The choice of the optimal r in each iteration could also be further improved by performing statistical analysis on the GAS algorithm's behaviour with different optimization problems. This way, a specific pattern could be proposed based on statistics and previous experience. Moreover, this could be more effective if some pre-conditioning of the cost function strategies is done to begin the algorithm in a condition where negative values are less than half of the solution space.

Even though the current status of the work is preliminary, the analysis performed to understand that the best setup for GAS effectiveness strictly depends on the problem to be solved and that an automatic toolchain can assist in properly configuring the QUBO solving procedure. Existing the relationship between the cost function profile and the GAS setup, a possible approach for the preliminary selection of all GAS features can be related to the coefficients of the QUBO matrix. For example, these can be exploited for identifying an initial offset of the cost function, obtaining a good initial ratio between positive and negative samples, and choosing the features of a QUBO problem classifier, which indicates the best GAS solver. Moreover, in the long-term perspective,



(a) Eight-node max-cut

(b) Seven-variable knapsack

(c) Four-variable garden optimization

(d) Four-variable nurse scheduling

Fig. 6: Cumulative distributions of the values obtained at the end of GAS, divided by the average total number of Grover rotations R. The complete statistical data for the considered problems are reported in Table I.

this toolchain could be expanded to support other quantum or quantum-inspired solvers tailored for each QUBO family, hoping that researchers can employ it to solve optimization problems having a social or industrial utility.

Appendix A
Grover Search Algorithm

Grover Search (GS) algorithm is a quantum routine for searching items in an unordered database, introducing a quadratic speed-up compared with its classical counterpart [16]. Its main idea consists in labelling the states encoding the solution of the search problem and then amplifying their measurement probability.

The conceptual scheme of GS is reported in the dashed box of Figure 4. In its typical formulation, given a database $\mathbb{D}$ and assuming an initial state $|0\rangle^{\otimes n}$ (with n total number of qubits), a unitary operator A for obtaining a uniform superposition of the states encoding $d \in \mathbb{D}$ is applied:

$$|\psi\rangle = A\,|0\rangle^{\otimes n} = \sum_{d\in\mathbb{D}} \frac{1}{\sqrt{\dim(\mathbb{D})}}\,|d\rangle \ . \tag{20}$$

Then, as reported in Equation 21, the Grover operator G is applied r times to the state $|\psi\rangle$ to perform the labelling and amplification of target states:

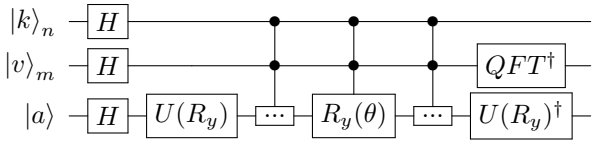
$$|\psi_f\rangle = (DO)^r\,|\psi\rangle = (G)^r\,|\psi\rangle \ , \tag{21}$$

where O and D are the unitary **oracle** and **diffusion** operators. The former flips the sign of the probability amplitude of target states, while the latter inverts all the amplitudes of the states around their average to amplify those of the target states. The complexity of GS increase as a $\mathcal{O}(\sqrt{\frac{N}{l}})$, where N possible states and l is the number of labeled states, as reported in [17]. In the GAS algorithm application, the unitary operator A is the quantum dictionary discussed in Section B. Moreover, since cost function values are written in two's complement representation, the oracle for labelling negative values can be obtained by applying a Z gate on the MSB of the qubits value register. Finally, D is constructed as $D = A \cdot \mathrm{C}^n Z_{|0\rangle^{\otimes n}} \cdot A^\dagger$, where $A^\dagger$ is the adjoint of A and $\mathrm{C}^n Z_{|0\rangle^{\otimes n}}$ is a multi-controlled-Z gate, flipping the sign of the probability amplitude of $|0\rangle^{\otimes n}$.

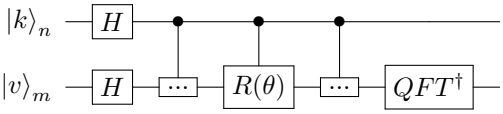
Appendix B
Quantum Dictionary

The **quantum dictionary** is a quantum circuit presented in [18], which encodes, equivalently to its classical counterpart, a key-value-pair data structure, such as a function, into a quantum state. It is implemented through two entangled quantum registers, one for the keys ($|k\rangle$ on n qubits) and the other for values ($|v\rangle$) on m qubits). Entanglement plays the role of a pointer in classical computation and assures correlation among each key and its corresponding value.

Different **encoding operators** can be used for defining a



(a) Implementation with the ancilla qubit, $U(R)$ prepares the state of the ancilla qubit, $R_y(\theta)$ is an element of the encoding operator.



(b) Implementation without the ancilla qubit, $R(\theta)$ is an element of the encoding operator.

Fig. 7: Circuits for quantum dictionary.

quantum dictionary. A popular approach derives from the **Quantum Phase Estimation** (**QPE**) [18] algorithm, in which a quantum state encodes the eigenvalue of an operator through its geometric series. At the circuital level, this approach can be implemented in two functionally equivalent ways, i.e. both approaches intrinsically produce the same final state, with keys associated with the combination of binary variables and values encoding those of the cost function, according to the two's complement representation.

The first approach exploits the unitary matrix $R_y(\theta)$ [19]; in particular, it requires an additional qubit over the $n + m$ required for the two registers, whose phase encodes the function values. In Figure 7a, the conceptual quantum circuit scheme is reported. The initialization step requires applying a set of Hadamard gates on all qubits to create a uniform superposition. Moreover, an additional sequence of single-qubit gates $U(R_y)$ is applied on the ancilla qubit to ensure that its state corresponds to an eigenvector of $R_y(\theta)$. Then, a sequence of controlled-$R_y(\theta)$ gates is applied to the ancilla qubit to describe the function values in a qubit phase geometric series, which can be finally converted into a single binary number on the value qubits through the **inverse Quantum Fourier Transform** (**QFT**†) [20]. Finally, uncomputation is done on the ancilla qubits to restore state $|0\rangle$. A slightly different approach avoids the ancilla qubit, applying to the value register (i.e. $|v\rangle$) a series of geometrically-spaced controlled rotation operators to encode the desired value in the phase of the state of $|v\rangle$. Also, in this case, it is possible to reconstruct the values on the computational basis by applying the inverse QFT on $|v\rangle$ qubits. Figure 7b reports an example circuit that implements a quantum dictionary with this technique. R practically corresponds to the $U_1(\theta)$ gate available in [11].

The second approach was applied to encode the cost function in the GAS implementation because it involves one qubit less, thus resulting less memory-intensive.

REFERENCES

[1] E. Zahedinejad and A. Zaribafyan, *Combinatorial Optimization on Gate Model Quantum Computers: A Survey*, 2017.

[2] V. S. Denchev et al., *What is the Computational Value of Finite-Range Tunneling?*, Phys. Rev. X, vol. 6, no. 3, p. 031015, Aug. 2016.

[3] E. Mounier, *Quantum Technologies: Market and Technology Report 2020*, 2020 [Online].

[4] C. Durr and P. Hoyer, *A Quantum Algorithm for Finding the Minimum*, arXiv, 1996.

[5] F. Glover, G. Kochenberger, and Y. Du, *A tutorial on formulating and using QUBO models*, arXiv 1811.11538, 2018.

[6] J. T. Iosue, *qubovert Documentation*, 2019 [Online].

[7] M. W. Coffey, *Adiabatic quantum computing solution of the knapsack problem*, arXiv 1701.05584, 2017.

[8] C. D. Gonzalez Calaza, D. Willsch, and K. Michielsen, *Garden optimization problems for benchmarking quantum annealers*, Quantum Information Processing, vol. 20, no. 9, pp. 1–22, 2021.

[9] K. Ikeda, Y. Nakamura, and T. S. Humble, *Application of quantum annealing to nurse scheduling problem*, Scientifc reports, vol. 9, no. 1, pp. 1–10, 2019.

[10] D. W. Bulger, W. Baritompa, and G. R. Wood, *Implementing Pure Adaptive Search with Grover's Quantum Algorithm*, Journal of Optimization Theory and Applications, vol. 116, pp. 517–529, 2003.

[11] *Qiskit Optimization GitHub repository*, [Online].

[12] A. Gilliam, S. Woerner, and C. Gonciulea, *Grover Adaptive Search for Constrained Polynomial Binary Optimization*, Apr. 2021.

[13] W. P. Baritompa, D. W. Bulger, and G. R. Wood, *Grover's Quantum Algorithm Applied to Global Optimization*, Jan. 2005.

[14] Y. Liu and G. J. Koehler, *Using modifcations to Grover's Search algorithm for quantum global optimization*, European Journal of Operational Research, vol. 207, no. 2, pp. 620–632, 2010.

[15] *Intel Xeon Gold 6134 Processor - Product Specifcation*. [Online]

[16] L. K. Grover, *Quantum Mechanics Helps in Searching for a Needle in a Haystack*, Phys. Rev. Lett., vol. 79, no. 2, pp. 325–328, Jul. 1997.

[17] S. Sadana, *Grover's search algorithm for n qubits with optimal number of iterations*, arXiv 2011.04051, 2020.

[18] A. Gilliam et al., *Foundational Patterns for Efficient Quantum Computing*, 2021.

[19] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, 2010

[20] D. Coppersmith, *An approximate Fourier transform useful in quantum factoring*, 2002, [Online].

Luigi Giuffrida received the B.Sc. and M.Sc. degrees in Electronic Engineering, in 2019 and 2021 respectively, from Politecnico di Torino. During the M.Sc. thesis he focused mainly on the study of Grover Adaptive Search and its degrees of freedom to efficiently solve QUBO problems in a quantum-classical mixed framework. He will attend the Ph.D. programme in Electrical, Electronics and Communications Engineering at Politecnico di Torino in November 2022. The main focus of his research activity will be the design of hardware architectures for complex computation, mainly targeting network processors, hardware for digital signal processing and neural networks.

Deborah Volpe (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in Electronic Engineering – in 2019 and 2021, respectively – from Politecnico di Torino, where she is now pursuing the Ph.D. degree in Electrical, Electronics and Communications Engineering. Her research interests mainly focus on the emulation of quantum computers on classical hardware (FPGA, CPU and GPU) and quantum-compliant approaches for solving QUBO problems.

Giovanni Amedeo Cirillo (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in Electronic Engineering and the Ph.D. degree in Electrical, Electronics and Communications Engineering from Politecnico di Torino in 2016, 2018 and 2022 respectively. He is currently a Digital Design Engineer in the Analog, MEMS & Sensors Group R&D team of STMicroelectronics (Cornaredo, Italy). His research activities are mainly related to device compact modelling for classical simulation of quantum computing and quantum-assisted communication technologies, quantum circuits design and optimization, definition of quantum-compliant algorithms for Telecommunications Industry and algorithmic-based design of digital signal processing architectures, mainly exploiting High-Level Synthesis (HLS) techniques. Between 2019 and 2021, he was the treasurer of the Politecnico di Torino IEEE Student Branch.

Maurizio Zamboni received the Degree in Electronics Engineering in 1983 and the Ph. D. degree in 1988 at the Politecnico di Torino. He joined the Electronics Department, Politecnico di Torino, in 1983, became Researcher in 1989, Associate Professor in 1992 and Full Professor of Electronics in 2005. His research activity started with the study of multiprocessor architectures, then he worked with digital IC design concentrating both on architectural aspects and circuits optimisation. In these years he got an expertise in the design of special ICs for Artificial Intelligence, Vision and Telecommunication. His main interests include now low-power circuits and innovative technologies beyond the CMOS world such as Nano Magnetic Logic and NanoArrays. From 2018 he started moving to Quantum Computing issues, contributing to the creation, at Politecnico of Torino, of a research group very active in many fields of the QC world, from technologies to emulators/simulators up to AI and ML applications. Many activities have been carried on from the device modeling for the promising technologies, up to the development of basic cells models for Quantum Gate Arrays and the study of applications of QC in the world of Communications, Security and AI. He is co-author of more than 200 scientific papers (three invited papers) and three books and holds two patents.

Giovanna Turvani received the M.Sc. degree with honours (Magna Cum Laude) in Electronic Engineering in 2012 and the Ph.D. degree from the Politecnico di Torino. She was Postdoctoral Research Associate at the Technical University of Munich in 2016. She is currently Assistant Professor at Politecnico di Torino. Her interests include CAD Tools development for non-CMOS nanocomputing, architectural design for field-coupled nanocomputing and high-level device modelling for Quantum Computing and hardware systems for microwave imaging-based techniques for biomedical applications and for food quality monitoring. Other expertise includes also the design of IoT low-power systems based on long-range protocols (LoRa).