Integration of Simulated Quantum Annealing in Parallel Tempering and Population Annealing for Heterogeneous-Profile QUBO Exploration

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Integration of Simulated Quantum Annealing in Parallel Tempering and Population Annealing for Heterogeneous-Profile QUBO Exploration

DEBORAH VOLPE[1], (Graduate Student Member, IEEE), GIOVANNI AMEDEO CIRILLO [1], (Member, IEEE), MAURIZIO ZAMBONI [1], and GIOVANNA TURVANI [1]

[1]Department of Electronics and Telecommunications of Politecnico di Torino, Torino, 10129, Italy

Corresponding author: Deborah Volpe (e-mail: deborah.volpe@polito.it).

ABSTRACT Simulated Quantum Annealing (SQA) is a heuristic algorithm which can solve Quadratic Unconstrained Binary Optimization (QUBO) problems by emulating the exploration of the solution space done by a quantum annealer. It mimics the quantum superposition and tunnelling effects through a set of correlated replicas of the spins system representing the problem to be solved and performing Monte Carlo steps. However, the effectiveness of SQA over a classical algorithm strictly depends on the cost/energy profile of the target problem. In fact, quantum annealing only performs well in exploring functions with high and narrow peaks, while classical annealing is better in overcoming flat and wide energy-profile barriers. Unfortunately, real-world problems have a heterogeneous solution space and the probability of success of each solver depends on the size of the energy profile region compatible with its exploration mechanism. Therefore, significant advantages could be obtained by exploiting hybrid solvers, which combine SQA and classical algorithms.

This work proposes four new quantum-classical algorithms: Simulated Quantum Parallel Tempering (SQPT), Simulated Quantum Population Annealing (SQPA), Simulated Quantum Parallel Tempering - Population Annealing v1 (SQPTPA1) and Simulated Quantum Parallel Tempering - Population Annealing v2 (SQPTPA2). They are obtained by combining SQA, Parallel Tempering (PT), and Population Annealing (PA). Their results are compared with those provided by SQA, considering benchmark QUBO problems, characterized by different profiles. Even though this work is preliminary, the obtained results are encouraging and prove hybrid solvers' potential in solving a generic optimization problem.

INDEX TERMS Simulated Quantum Annealing, Parallel Tempering, Population Annealing, Hybrid Quantum-Classical Algorithms, Optimization Problems, Quadratic Unconstrained Binary Optimization, Ising machine, Cost Function, Energy profile.

I. INTRODUCTION

OPTIMIZATION target is finding a variableconfiguration that **minimizes a cost function or maximizes a fitness one**. It is relevant in many real-world applications, such as chemical simulations [1], logical and physical VLSI circuit synthesis [2], resource allocation in industrial environments [3], structural optimization [4] [5], antenna design [6], and many others. It can be subdivided into two categories, depending on the variables' nature: discrete or **combinatorial optimization** (**CO**), in which variables are discrete, and continuous optimization, in which the same are continuous.

For some applications, not all solutions are feasible and constraints must be applied to variable (**constrained optimization**). They can be taken into account differently depending on the solvers' characteristics: inserting them inside the objective function (through **penalty functions**) and directly generating only feasible new configurations during the exploration.

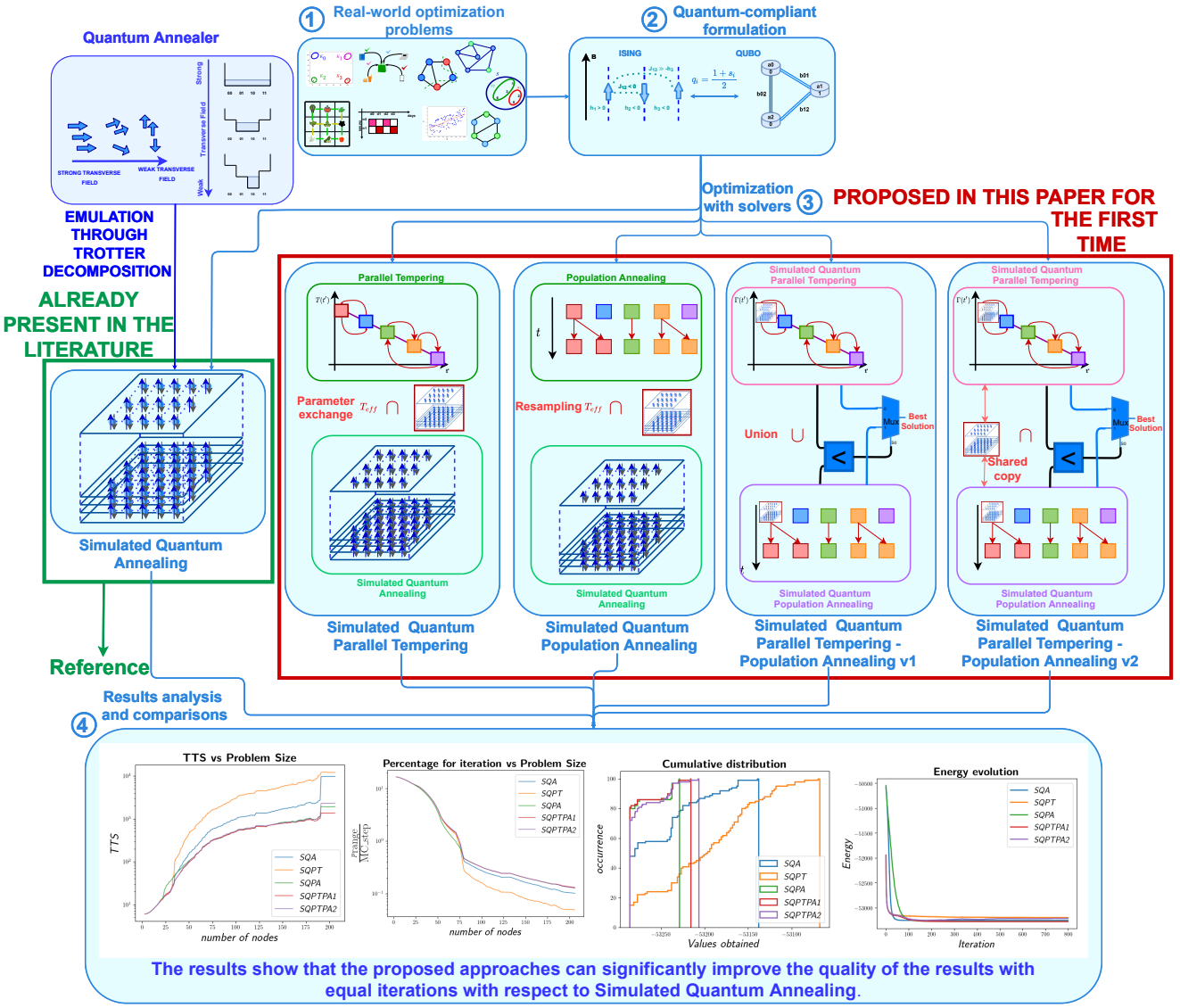


FIGURE 1: Overview of the article. Starting from the Simulated Quantum Annealing (SQA) algorithm for the emulation of quantum annealers, different algorithms for solving Ising/QUBO problems — based on interfacing SQA with Parallel Tempering and Population Annealing — have been defined. Comparisons between the proposed methodologies and that are already present in the literature were done for different real-world optimization problems describable with Ising/QUBO formulation. Reminding that these algorithms are all iterative, the obtained results prove that, with the same number of iterations, the proposed approaches can improve the quality of the obtained results with respect to SQA.

Moreover, in a real-world context, it could be necessary to optimize more than one objective (fitness or cost function) at the same time. In this case, it is called **multi-objective or vector optimization** (**MO**), and, in this case, the solution is always a trade-off between functions involved in the problem. The most preferred solution can be found by computing a representative set of **Pareto optimal solutions** (impartial approach), which a human decision marker can evaluate during (interactive methods) or at the end of optimization (*a posteriori* methods) expressing preferences, or by combining objective functions into a higher **scalar** one (only one objective function involved) through **aggregation approaches** (*a priori* methods), which exploit a preference criterion.

Several strategies were proposed for solving optimization, and the best one commonly depends on the optimization class.

A brute-force approach, i.e. the analysis of all possible input variables combinations, always guarantees the achievement of the optimal solution. However, the execution time increases exponentially with the problem size, thus making it unfeasible in a real-world context.

Deterministic explorations can be grouped into three cat-

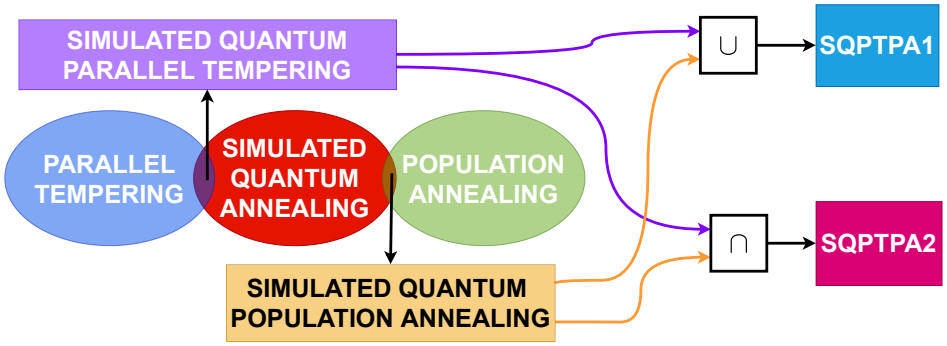


FIGURE 2: Relationship between Simulated Quantum Annealing, Parallel Tempering, Population Annealing and the proposed algorithms.

egories: second-order methods (such as Newton's one [7]), which are based on both gradient and hessian computation, first-order methods (such as quasi-Newton one [8]), which exploit only the gradient, and zero-order methods (such as pattern searched [9]), which use only the objective function. They are effective only for some optimization problems, e.g. the convex ones, which are characterized by the presence of a single minimum (or maximum). Moreover, they can have a significant latency for reaching convergence.

Therefore, **heuristic approaches** are currently the best way to find optimal or sub-optimal solutions for large-scale optimization problems. They are commonly inspired by natural phenomena and based on an equilibrium between the **exploration** (divergence) of the feasible solutions space and the **exploitation** (convergence) of the knowledge acquired evaluating the previously obtained solutions. Furthermore, several famous heuristic algorithms are **population-based**, i.e. there are a set of elements (individuals) which explore the solution space in parallel with specific rules deriving from some natural or social phenomena.

Some of them emulate the behaviours of a set of individuals or animals with the same target. They are also called **cooperative algorithms**. The most famous algorithms in this context are the **particle swarm** [10], which is based on the behaviour of a flock of birds (**candidate solutions**) looking for food (motion of the candidate solutions in the objective function space for finding the optimum), and the **ant colony** [11], which mimics the cooperative behaviour of ants that communicate where is food. In the first one, each element must follow its neighbours, stay in the flock and avoid collisions. The exploration is guaranteed by the possibility for an individual to get out of the flock if a promising region in the solution space is found (selfish behaviour), while exploitation is associated with the tendency to stay in the flock (social behaviour). At the same time, in the second algorithm, ants randomly explore the area surrounding their nest (exploration). When an ant finds a promising region, it attracts others in the same area (exploitation).

In recent years, many other cooperative population-based approaches have been proposed, for example, **teaching-learning** optimization [12], [13], which mimics the learning process of a class. In particular, there is a class of students, whose scores are the quality of the associated solution. The individual associated with the best solution is the teacher. The learning process of each element depends on both the classmates' and teacher's experience (exploitation).

Other population-based algorithms mimic the biological evolution of a species, which modifies its characteristic to better survive in the environment. They are also called **evolutionary algorithms** and are based on the competition among individuals in the population. The oldest and most popular in this field are the **evolutionary strategy** [14] and the **genetic algorithms** [15]. In the first, a population of individuals (configurations) is generated. Then, their ability is measured (quality of the solution) and a new generation of individuals is generated by selecting in a deterministic way the best individuals, associated with the best current solutions, by inheriting the best characteristics of the previous one (genetic operation), i.e. combining two or more individuals (exploitation), and also mimicking the genetic mutation, corresponding to random events due to errors in the copy operation (exploration). The second one is based on the same principles as the previous, but with the following differences: the variable representation is binary instead of a floating-point one, which helps to implement a more complex individuals combination mechanism and easier implementation of mutation, the selection mechanism is probabilistic instead of deterministic and the genetic operation is fixed instead of changing during the solution space exploration.

Also, in this context, many other evolutionary population-based approaches have been proposed, in recent years. For example, the **follow-the-leader** approach [16] is based on the behaviour of a sheep within a flock foraging. In particular, the sheep in the region with more green grass around (the current best solution) is identified as the leader of the flock, while the one with less or dry grass around (worst current solution) is the rear one. The flock tends to follow the leader, so this phenomenon corresponds to moving the search to the most promising region (exploitation of the knowledge). The main issue of the proposed original algorithm is that there is

an unbalance between exploitation and exploration, which can compromise the quality of the results in non-convex (multi-modal) optimization problems. The work presented in [17] tries to overcome this limitation, thus allowing the algorithm to express its potential fully.

The last historical type of population-based algorithm is the **artificial immune** one [18], which emulates a living body's defence from external biological enemies that change with time. In these algorithms, the objective function plays the role of antigen, i.e. a substance able to stimulate the immune system response, while the antibodies represent the feasible solution, defining a binary code for each one. In the beginning, an initial antibody population is generated. The quality of each element is evaluated in terms of affinity among antigens and antibodies. According to this value, the antibodies are selected for cloning (exploitation) and a mutation mechanism is applied (exploration). The perspective and recent proposal of these approaches are summarized in [19].

Furthermore, many popular heuristic algorithms are inspired by the evolution of a physical system. The most iconic example is **simulated annealing** (SA) [20]. As the name suggests, it mimics physical annealing, which is a process exploited in materials science for removing reticular defects of crystals based on two steps. The first one is heating crystals over their re-crystallization temperature, to allow the motion of the atoms in the lattice, the second one is slow cooling of the same, to achieve the optimal settling of atoms in the lattice. SA implements the exploration by accepting with certain probability — depending on the temperature parameter (**Metropolis-Hastings** approach, explained in Section II-B) — new solutions that can degrade the objective function's current value. On the other hand, exploitation is done by adopting a neighbour exploration policy and by the systematic acceptance of better new solutions. The convergence is guaranteed by the gradual reduction of the temperature parameter, i.e. the probability of accepting a degrading solution. The interest in this approach is grown over the years: several approaches for improving it have been proposed, e.g. **parallel tempering** and **population annealing** (explained in Section II), it is currently employed in many applications [21], [22], [23], [24], [25], and it has been exploited for improving the balance between exploration and exploitation of more complex optimization mechanisms [26], [27], [28].

Another algorithm exploiting thermodynamic laws is that presented in [29]. This emulates the behaviour of a system for reaching thermal equilibrium. It is also a population-based approach, where each molecule of the system is an element of the population. In particular, the simulated physical phenomena are conduction, convection and radiation.

In the same context, another recent algorithm is **plasma generation optimization** [4]. As the name suggests, it is inspired by the plasma generation process, in which the electrons play the role of elements of a population, and the exploration of the solution space is performed by emulating excitation modes, de-excitation and ionization processes.

The presence of so many approaches in the state-of-art proves that non-convex optimization is a crucial task in many fields of application. Moreover, the available methods are sometimes **not entirely satisfactory** in terms of the quality of the results or the time required. This is also due to the fact that **the effectiveness of an exploration mechanism strongly depends on the characteristics of the optimization problem of interest**. For example, SA cannot effectively explore an objective function with high and narrow peaks, while is particularly efficient in exploring flat and wide regions. For this reason, it is possible to conclude that, if the characteristics of a non-convex optimization problem are unknown or unpredictable, combining multiple algorithms effective on complementary types of problems could improve the mean quality of the obtained solutions.

A. RESEARCH GAP AND MOTIVATIONS OF THE STUDY

Interest in exploiting quantum computing for optimization problems has grown in the last two decades to try to overcome the limitations of the currently available classical algorithms. In particular, the exploitation of quantum mechanics principles like superposition, entanglement and tunneling can help to define algorithms for data-intensive applications with lower computational complexity; in the optimization context, quantum procedures can achieve a good compromise between solution quality, execution time and computational complexity.

The most feasible formulations, introduced in Section II, for solving CO with quantum-compliant approaches are **Ising** and **Quadratic Unconstrained Binary Optimization** (**QUBO**). There are two possibilities for exploiting the quantum computing paradigm in the optimization context: exploiting a **quantum annealer** — which is a **special-purpose quantum computer** theorized in 1998 [30], [31], [32] and exploiting the natural properties of a quantum system to minimize a cost function (detailed in Section II-B) — and proposing algorithms to be entirely or partially executed on a **general-purpose quantum computer compliant with the quantum circuit model** [33]. In the second context, quantum computers are usually employed as sub-routines for the acceleration of specific tasks of more complex algorithms also involving classical computers. The **Grover Adaptive Search** [34], [35] algorithm well describes this mechanism. It is a successive approximation algorithm exploiting the well-known quantum Grover's search algorithm to find the negative values of a cost function that is iteratively classically moved up of the value of the last negative sample obtained. Other quantum-classical algorithms for solving optimization problems are those based on variational routines like the ones in [36] [37] or the quantum-genetic presented in [38].

Nevertheless, quantum devices are currently in the so-called **Noisy Intermediate-Scale Quantum** (NISQ) era [39]. These are characterized by a very limited amount of qubits, with limited connectivity and are subjected to non-ideality phenomena. These conditions could make unfeasible their current employment in real-world applications.

Consequently, the interest in quantum-inspired and quantum emulation algorithms grows in the last years for exploiting the quantum principles with classical devices and several approaches have been proposed, e.g. **Simulated Quantum Annealing** (explained in Section II), **simulated adiabatic bifurcation** [40] and **digital annealing** [41]. In particular, **Simulated Quantum Annealing** (**SQA**) mimics the solution space exploration of a quantum annealer on digital hardware by using **quantum Monte Carlo simulation**. It was proven that SQA provides, for some types of problems, an **exponential speed-up with respect to classical simulated annealing** [42]. In particular, it provides a significant **advantage for problems with high and narrow peaks in the objective function** (spike problems).
However, real-world problems are **heterogeneous**, i.e. composed of flat and wide regions and high and narrow barriers. Consequently, new methods for exploring effectively this kind of objective functions are required. As deeply explained in Section III and in [43], a significant advantage could be obtained with **hybrid solvers**, which can efficiently alternate classical annealing (local search) and the simulated quantum one (global search). In this way, the best of both exploration mechanisms is taken.
As a result, this work is going to propose four new algorithms: **Simulated Quantum Parallel Tempering** (**SQPT**), **Simulated Quantum Population Annealing** (**SQPA**), **Simulated Quantum Parallel Tempering - Population Annealing v1** (**SQPTPA1**) and **Simulated Quantum Parallel Tempering - Population Annealing v2** (**SQPTPA2**). They combine Parallel Tempering, Population Annealing and SQA, as shown in Figure 2, for exploring effectively heterogeneous energy profile. These algorithms are compared with SQA to prove their effectiveness and efficiency in solving different CO problems. The identification of the best strategy for each problem represents a milestone for developing an automatic toolchain for improving QUBO solving.
Therefore, this work contributes to the research as follows:

- adapts the effective temperature equation proposed in [43] for the quantum annealer to the SQA;
- determines the meaning of a system copy for SQA;
- proposes Simulated Quantum Parallel Tempering algorithm, which combines Parallel Tempering and SQA, exploiting the obtained effective temperature equation and the identified system copy concept;
- proposes Simulated Quantum Population Annealing algorithm, which combines Population Annealing and SQA, exploiting the obtained effective temperature equation and the identified system copy concept;
- proposes Simulated Quantum Parallel Tempering - Population Annealing v1, which is the *union* of Simulated Quantum Parallel Tempering and Simulated Quantum Population Annealing;
- proposes Simulated Quantum Parallel Tempering - Population Annealing v1, which is the *itersection* of Simulated Quantum Parallel Tempering and Simulated Quantum Population Annealing, defining a strategy to manage a shared system copy among the two algorithms;
- tests the proposed algorithms with nine different families of optimization problems and compares the results with ones of SQA;
- proves that the best strategy is problem-dependent and that the proposed approaches can significantly improve the quality of the results with respect to SQA.

The manuscript content is summarized in the graphical abstract reported in Figure 1.

The article is organized as follows. Section II reports theoretical foundations, particularly the QUBO and Ising formulation, the considered benchmark problems and an explanation of Simulated Quantum Annealing, Parallel Tempering and Population Annealing. Section III reports the proposed algorithms, motivating the necessity of hybrid solvers. In Section IV, the results are reported and discussed. Finally, in Section V, conclusions are drawn, and future perspectives are illustrated.

II. THEORETICAL FOUNDATION

A. OPTIMIZATION PROBLEMS FORMALISM

As mentioned, the most feasible formulations for solving optimization problems with quantum approaches are the **Quadratic Unconstrained Binary Optimization** (**QUBO**) and the **Ising** ones, which are introduced in the following paragraphs. The two models are strongly **correlated**, and it is always possible to move from one to the other, exploiting the relations reported in Paragraph II-A3.

1) QUBO formalism

Quadratic Unconstrained Binary Optimization (**QUBO**) is a mathematical formulation capable of describing many real-world problems [44], such as placement [45], [46], routing [47] and scheduling [48]. It involves unipolar binary variables, i.e. which can assume only 0 and 1 values, as the terms **Binary** in the acronym suggest. Therefore, it can only describe CO problems. The **Quadratic** term refers instead to the highest power applied to them, **Unconstrained** indicates that the variable constraints cannot be explicitly taken into account, and **Optimization** puts in evidence that this model is exploited for minimizing or maximizing the obtained objective function, which can be written as:

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

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, b_{ij} is a strength which controls the influence of variables i, and j and c is an offset, which can be neglected during the optimization.
It can also be expressed as:

$$\text{minimize/maximize} \quad y = \mathbf{x}^t \cdot Q \cdot \mathbf{x} \,, \quad (2)$$

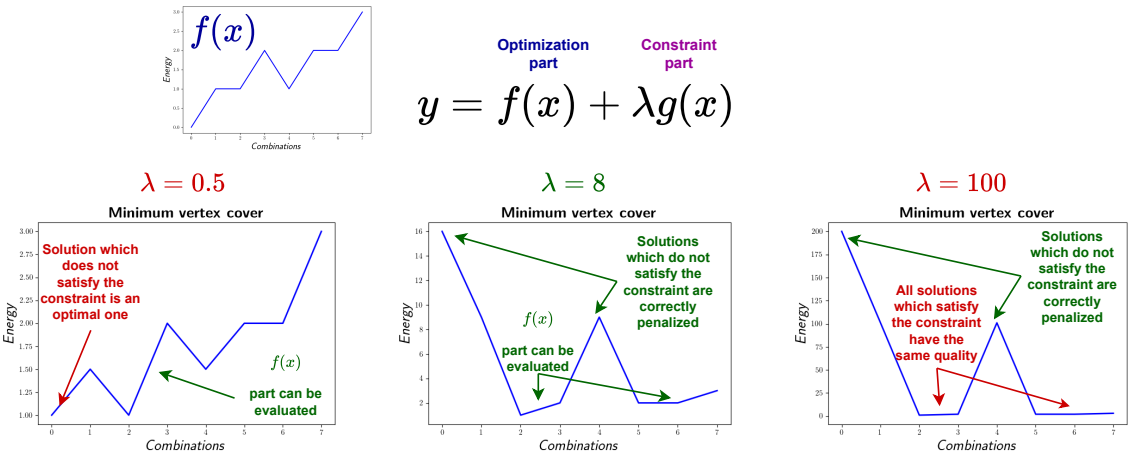


FIGURE 3: Influence of λ value on the solution quality. A too-low value (on the left) could give unreliable solutions because ones which do not satisfy the constraint are not correctly penalized with respect to the other. While a too-high value makes all solutions that satisfy the constraints equal from an energy point of view, thus implying that it is impossible to distinguish the optimal one for the $f(x)$ function.

where **x** is a vector of binary variables (*e.g.* [0,1,1,0,1]) and Q is a square matrix of constants, depending on the problem. The matrix Q can be **symmetric** or in **upper triangular form**.

Despite what the name would suggest, the variable constraints can be taken into account by introducing quadratic penalties to the objective function:

$$\text{minimize/maximize} \quad y = f(\mathbf{x}) + \lambda g(\mathbf{x}) \,, \tag{3}$$

where λ is a positive penalty parameter to be multiplied by the constraint function or **penalty function** $g(\mathbf{x})$. In this way, the constraints are evaluated during the optimizer execution. Nevertheless, sizing the variable λ is crucial. Indeed, with a too-low value, the constraint could be neglected, thus implying the unreliability of the obtained solutions. While a too-high value makes the objective function too flat, complicating the evaluation of the effective quality of the solution, as shown in Figure 3. Tutorials [44] suggest taking λ as a certain percentage of the original objective function (usually in the range **75%-150%**).

Multi-objective optimization can be also performed considering QUBO formulation by exploiting an aggregation approach (**Objective Weighting**), which combines objectives into a single one as explained in [49].

2) Ising formalism

The **Ising** model [50], [51] is a physical-mathematical model of **ferromagnetism** used in statistical mechanics. It consists of a system of **atomic spins** described as **dipoles**, each of which can be in one among two discrete states +1 (spin-up) or -1 (spin-down), depending on its orientation. Spins are arranged in a lattice allowing each spin to interact with its neighbours. The following Hamiltonian can describe this model:

$$H(\mathbf{s}) = \frac{1}{2} \sum_{i=0}^{N-1} \sum_{j=0, j \neq i}^{N-1} J_{ij} s_i s_j + \sum_{i=0}^{N-1} h_i s_i \,, \tag{4}$$

where N is the number of spins, s_i is the i^{th} spin, J_{ij} is the interaction coefficient among the i^{th} and the j^{th} spins, which is equal to the coefficient J_{ji} (a symmetric format is considered for the interaction coefficients matrix J), and h_i is the external magnetic field coefficient of the i^{th} spin. Therefore, the Hamiltonian includes two types of interaction:

- **External field** h, whose sign determines if the spin prefers up or down orientation. The size of h represents the weight of the energy contribution of a single spin with respect to the others.
- **Interaction terms** J between neighbor spin pairs. J gives the weight of the coupling and the sign indicates if neighbours prefer aligned or anti-aligned (more properly ferromagnetic and anti-ferromagnetic) orientation. Each pair contribution has to be summed to obtain the overall interaction energy.

This model can be exploited to describe many types of systems, where each spin is associated with an involved element, which can be paired with the others.

Ising model can be **1D**, **2D**, **3D**, or **fully-connected** depending on the number of interacting neighbours for each spin (Figure 4). In a 1D structure, each spin has two neighbours; in 2D, it has four neighbours; in 3D, it has six ones; in a fully-connected structure, each spin interacts with all the others. The last one is not feasible in a physical system, but it is a useful theoretical extension which permits expressing any problem with very high flexibility.

CO problems can be mapped onto the Ising model so that the **ground state** corresponds to their optimal solution. Ground-state search is then executed by updating spins stochastically.

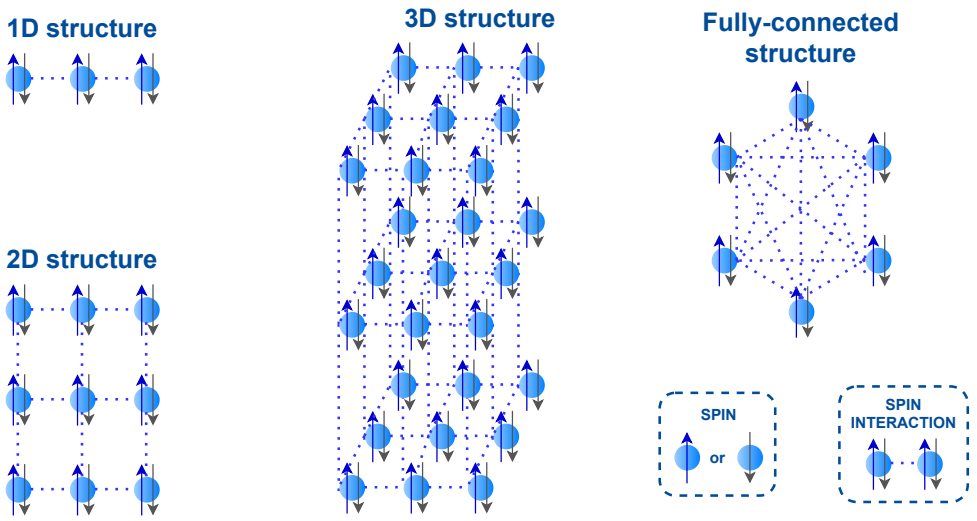


FIGURE 4: 1D, 2D, 3D and fully-connected Ising structures. In a 1D structure, each spin has two neighbours; in 2D, it has four neighbours; in 3D, it has six ones; in a fully-connected structure, each spin interacts with all the others. It is a virtual expansion of the physical model for describing any optimization problems with very high flexibility.

This update is performed by some algorithms like simulated or quantum annealing.

3) QUBO-Ising relation

Ising and QUBO models are perfectly equivalent [51], and it is always possible to move a problem from one formulation to another, by exploiting the following relation:

$$q_i = \frac{1 + s_i}{2}, \tag{5}$$

and its counterpart:

$$s_i = 2q_i - 1. \tag{6}$$

The only difference between the two models is related to the involved coefficients.

The formulation and this conversion can be assisted by Python libraries, such as qubovert [52], PyQUBO [53], [54] and dimod [55], characterized by routines for automatic insertion of some relevant constraints in the problem function and by the possibility of interfacing the defined QUBO or Ising problems with different solvers, *e.g.* based on simulated annealing [20].

4) Benchmark problems considered

This paragraph presents the problems considered for **benchmarking** the proposed solvers, which are graphically described in Figure 5. They were chosen because they differ significantly, especially regarding the energy profile, as shown in Figure 6. It was chosen to report the objective function because it is proven [43] that the effectiveness of quantum-annealing-based exploration strictly depends on the presence of **high** and **narrow peaks**, which profits from the tunnelling effect, as explained in Section III.

Each benchmark problem was implemented in QUBO formulation and converted in Ising one through the **qubovert** library.

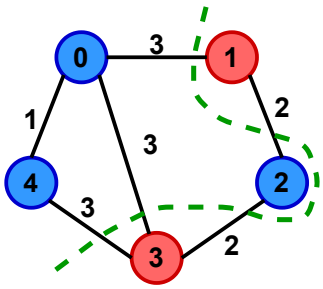
a: *Maxcut*

Maxcut [56] [57] (Figure 5a) is one of the most relevant CO problems, whose target 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**. This can be exploited to describe **several real-world problems** in network design, statistical physics, VLSI design and circuit layout design [58]. Its QUBO formulation requires **a binary variable for each node**, whose value is 1 or 0, depending on the subset to which the node belongs. A **cut** can be seen as **severing edges joining two sets**, and consequently, the quantity $\epsilon_{(i,j)} = x_i + x_j - 2x_i x_j$ recognizes whether the edge (i, j) is in the cut. In particular, the edge (i, j) is in the cut if $\epsilon_{(i,j)} = 1$, a condition taking place only if $x_i \neq x_j$. When x_i and x_j are both equal to one or equal to zero, $\epsilon_{(i,j)} = 0$ Considering the contributions of each edge, the following objective function is obtained:

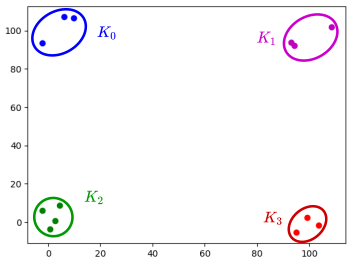
$$\begin{aligned} \text{Maximize} \quad y &= \sum_{(i,j)\in E} w_{i,j}\epsilon_{(i,j)} \\ &= \sum_{(i,j)\in E} w_{i,j} \cdot (x_i + x_j - 2x_i x_j), \end{aligned} \tag{7}$$

where $w_{i,j}$ is the weight of the edge that connects the i^{th} and the j^{th} node.

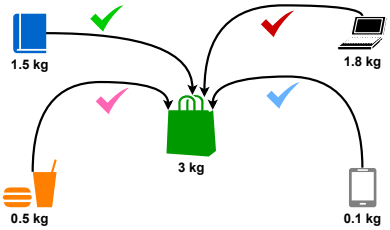
A peculiarity of maxcut problem regards its energy profile is **symmetric** (as shown in Figure 6a). In fact, a solution and its complement (e.g. [0,1,1,0,1] and [1,0,0,1,0]) have the same



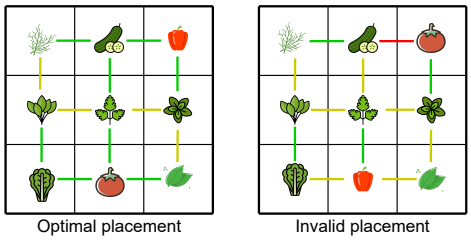
(a) Five-node max-cut



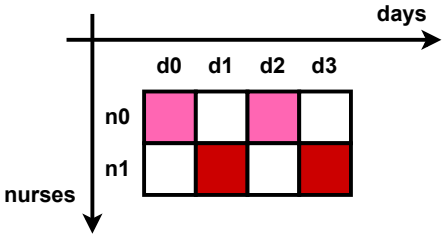
(b) Four-cluster thirteen-data clustering



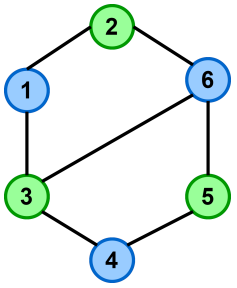
(c) Four-object knapsack



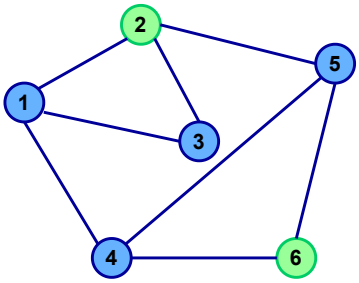
(d) Nine-pot nine-plant garden optimization



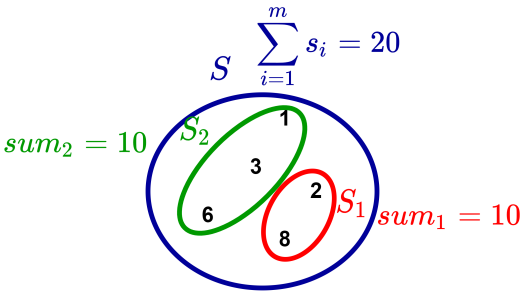
(e) Two nurse four-day nurse scheduling



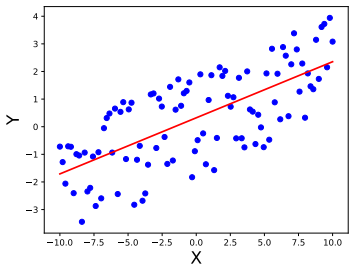
(f) Six-node two-color graph colouring



(g) Five-node minimum vertex cover



(h) Five-number partitioning



(i) Linear regression

FIGURE 5: Optimization problems considered for benchmarking.

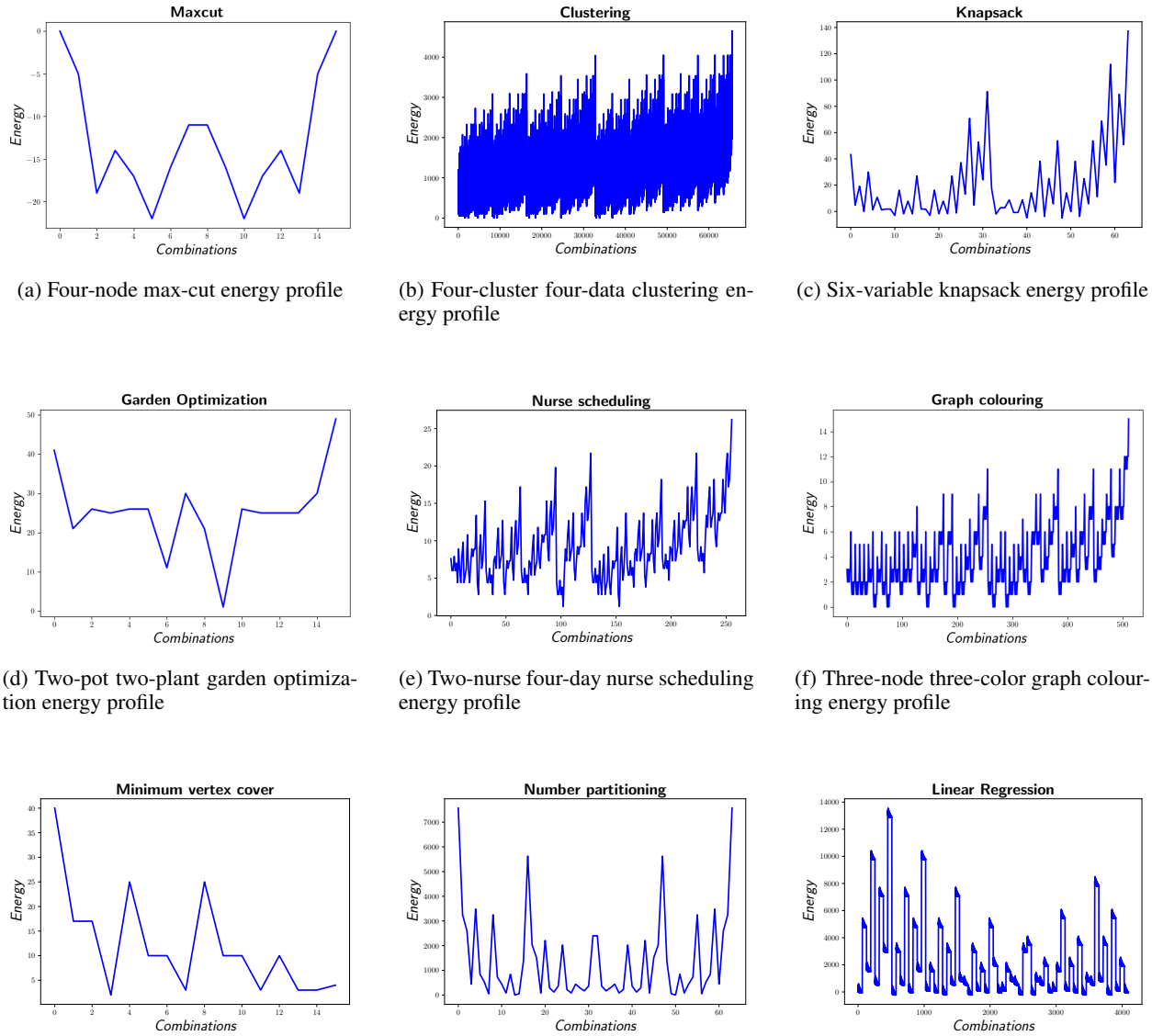
energy because the obtained two subsets are interchangeable.

b: *Data Clustering*

Clustering is an essential task of **unsupervised learning**, and its goal is **to divide a group of N data into K subgroups (cluster) of similar elements**. Similarity criteria can be associated with objective functions of optimization problems. Figure 5b shows a four-subgroup clustering of thirteen data, where closer data belong to the same cluster. The data distance is employed as a similarity metric to be minimized.

In this article, we considered the simplest data clustering QUBO formulation, which involves $N \cdot K$ **binary variables**, one **for each data-cluster pair**. The x_{nk} variable assumes value one if the n^{th} data is in the k^{th} cluster. Some requirements, which partially depend on the application, have to be satisfied for obtaining valid results. First of all, each data can be assigned to exactly one cluster:

$$\forall n: \quad \sum_{k=1}^{K} x_{nk} = 1\,. \tag{8}$$



(a) Four-node max-cut energy profile

(b) Four-cluster four-data clustering energy profile

(c) Six-variable knapsack energy profile

(d) Two-pot two-plant garden optimization energy profile

(e) Two-nurse four-day nurse scheduling energy profile

(f) Three-node three-color graph colouring energy profile

(g) Four-node minimum vertex cover energy profile

(h) Six-number partitioning energy profile

(i) Three-power and one hundred point linear regression energy profile

FIGURE 6: Considered optimization problems energy profiles.

In the considered case, data must be equally distributed among the clusters:

$$\forall k: \quad \sum_{n=1}^{N} x_{nk} = \frac{N}{K}\,. \tag{9}$$

Finally, the optimization figure of merit is the minimization of the total distance D among data in all clusters:

$$D = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K} d_{ij}x_{ik}x_{jk}\,, \tag{10}$$

where the distance between the i^{th} and the j^{th} data is computed as:

$$d_{ij} = \sqrt{(c1_i - c1_j)^2 + (c2_i - c2_j)^2 \ldots}\,, \tag{11}$$

where $c1, c2 \ldots$ are the coordinates (or features) associated with each data.

Summing all the contributions, the final objective function

can be written as:

$$f_{\text{cluser}}(\mathbf{x}) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K} d_{ij}x_{ik}x_{jk} + \lambda_1 \sum_{n=1}^{N}\left(\sum_{k=1}^{K} x_{nk} - 1\right)^2 + \\ + \lambda_2 \sum_{k=1}^{K}\left(\sum_{n=1}^{N} x_{nk} - \frac{N}{K}\right)^2 . \quad (12)$$

In Figure 6b an example of a problem energy profile is reported. It is possible to notice that several high peaks are present, implying that quantum exploration is effective for this application.

c: *Knapsack*

The **knapsack** problem [59] (Figure 5c) target consists in **defining for a set of objects** X, each of which is labelled as x_i and is characterized by a weight w_i, **the best subset to be put into a bag**, guaranteeing that the total weight does not exceed a threshold W:

$$0 < \sum_{i=1}^{\dim(X)} w_i x_i \leq W \,, \quad (13)$$

while maximizing the total preference score:

$$P = \sum_{i \in \text{subset}} p_i \,, \quad (14)$$

where p_i is exploited for expressing the preference of the i^{th} object, which is higher for an object more preferred to be put in the bag. When the terms p_i are all equal, the problem requires maximizing the total number of objects inserted in the bag, considering the weight limitation.
Since a **strict inequality** has to be expressed, QUBO formulation exploits auxiliary variables [44], [60] whose number depends on W value and can be partially limited by involving integer weights instead of real ones. Indeed, representing real numbers requires a more complex, e.g. the floating-point one.
The final objective function can be written as follows:

$$f_{\text{knapsack}}(\mathbf{x}) = f_{\text{inequality}}(\mathbf{x}) - \sum_i p_i x_i \,. \quad (15)$$

The formulation is sufficiently generic to be arranged for optimization problems concerning resource selection, e.g. in industrial environments.
An example of a knapsack problem energy profile is reported in Figure 6c.

d: *Garden optimization*

An important category of optimization problems is **optimal placement**. In this context, an interesting real-world application is **garden optimization** [61], whose target is **to optimally place** n **plants in** n **pots** (as shown in Figure 5d). This problem can be written in QUBO formulation with n^2 **binary variables**, one for **each plant-pot pair**. The x_{ij} variable assumes value one if the plant j is in pot i.
Some requests have to be satisfied to obtain a valid placement. First of all, every available plant must be placed in the garden:

$$\forall j: \quad \sum_{i=1}^{n} x_{ij} = 1 \,. \quad (16)$$

Then, each pot has to be filled with exactly one plant:

$$\forall i: \quad \sum_{j=1}^{n} x_{ij} = 1 \,. \quad (17)$$

Finally, tall plants shall not shadow smaller ones:

$$\forall i,j: \quad (i \mod 2 - s_j)^2 x_{i,j} = 0, \quad (18)$$

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

$$f_{\text{garden}}(\mathbf{x}) = -\sum_{i,i'=1}^{n} J_{ii'}\left(1 + \sum_{j,j'=1}^{n} x_{ij}C_{jj'}x_{i'j'}\right) + \\ + \lambda_1 \sum_{i=1}^{n}\left(1 - \sum_{j=1}^{n} x_{ij}\right)^2 + \lambda_2 \sum_{j=1}^{n}\left(1 - \sum_{i=1}^{n} x_{ij}\right)^2 + \\ + \lambda_3 \sum_{i=1}^{n}\sum_{j=1}^{n}(i \mod 2 - s_j)^2 x_{ij}, \quad (19)$$

where $C_{jj'}$ and $J_{ii'}$ are the terms of the companions C and adjacency J matrices, respectively. $C_{jj'}$ can assume values +1, 0 or -1, depending on the antagonist, a neutral or positive relationship among plants j and j', while $J_{ii'}$ is equal to 1 if pots i and i' are adjacent.
In Figure 6d, an example of a problem energy profile is reported.

e: *Nurse scheduling*

Another important category of optimization problems is **scheduling**. A symbolic application example is **nurse scheduling optimization** [48], which aims to find the **optimal assignment for nurses working in a hospital over a fixed timetable of shifts** (Figure 5e).
Considering N nurses and D working days, the QUBO formulation involves $N \cdot D$ **binary variables**, one for **each nurse-day pair**, which assumes value 1 if the n^{th} nurse works on the d^{th} day. A valid schedule must satisfy three conditions. The first one is called **hard nurse constraint** and ensures that **no nurse works for two consecutive days**. This is expressed by exploiting a positive correlation constant a, which penalizes the schedule for two consecutive days of the same nurse:

$$f_{\text{nurse}}(\mathbf{x}) = \sum_{n=1}^{N}\sum_{d=1}^{D-1} a \cdot x_{n,d} \cdot x_{n,d+1} \,. \quad (20)$$

The second one is called **hard shift constraint** and assures that in **each day** d **the nurse effort** $\sum_{n=1}^{N} E(n)x_{n,d}$ **is sufficient to satisfy the associated workload** $W(d)$:

$$\forall d: \quad \sum_{n=1}^{N} E(n)x_{n,d} = W(d)\,. \tag{21}$$

Finally, the **soft nurse constraint** assures that **all nurses should work approximately the same number of days** $F = D/N$:

$$\forall n: \quad \sum_{d=1}^{D} x_{n,d} = F\,. \tag{22}$$

The final objective function can be written as follows:

$$f_{\text{nurse}}(\mathbf{x}) = \sum_{n=1}^{N}\sum_{d=1}^{D-1} a x_{n,d}x_{n,d+1} + \lambda_1 \sum_{n=1}^{N}\left(\sum_{d=1}^{D} x_{n,d} - F\right)^2 + \\ + \lambda_2 \sum_{d=1}^{D}\left(\sum_{n=1}^{N} E(n)x_{n,d} - W(d)\right)^2. \tag{23}$$

In Figure 6e an example of a problem energy profile is reported.

f: *Graph colouring*

Graph colouring [44] is an optimization problem aiming to **assign different colour labels to adjacent nodes**, as shown in Figure 5f. This can be exploited in a wide range of applications in both industrial and scientific fields, e.g. printed circuit design [62]. Given K colours and N nodes, the associated QUBO formulation involves $N \cdot K$ **binary variables**, i.e. one for **each node-colour pair**, which assumes value one if the k^{th} colour is assigned to the n^{th} node. Some requirements have to be satisfied to obtain a valid solution.
First of all, adjacent nodes have to be assigned different colours:

$$\forall k: \quad \forall (i,j) \quad \text{adjacent node}: \quad x_{ik} + x_{jk} \leq 1\,, \tag{24}$$

which can be expressed as:

$$\forall k: \quad \sum_{i,j\in E} x_{ik}x_{jk} = 0\,, \tag{25}$$

where E is the set of edges of the graph.
Moreover, each node has to be assigned exactly one colour:

$$\forall n: \quad \sum_{k=1}^{K} x_{nk} = 1\,. \tag{26}$$

Finally, the objective function can be written as follows:

$$f_{\text{colouring}}(\mathbf{x}) = \lambda_1 \sum_{n=1}^{N}\left(\sum_{k=1}^{K} x_{nk} - 1\right)^2 + \lambda_2 \sum_{k=1}^{K}\sum_{i,j\in E} x_{ik}x_{jk}\,. \tag{27}$$

In Figure 6f an example of a problem energy profile is reported.

g: *Minimum vertex cover*

Minimum vertex cover [44] is an optimization problem involving an undirect graph with a set of vertices V and edges E. A **vertex cover** is the **subset of vertices such that each edge is incident to at least one vertex of the subset itself**. Consequently, the problem goal is to **find the cover with the minimum number of vertices in the subset**. For example, in Figure 5g, light blue nodes form a vertex cover because each edge of the considered graph is connected to at least one of these. It is also the one with the lower number of involved nodes, i.e. the optimal solution. Considering N vertices, the QUBO formulation involves N **binary variables**, one **for each vertex** x_i, whose value is one if the node is in the cover, i.e. in the subset. The final objective function is composed of two parts. The first one minimizes the number of nodes in the subset:

$$\text{Minimize} \quad y_1 = \sum_{i\in V} x_i\,. \tag{28}$$

The second one assures that each edge is incident to at least one vertex in the subset, forcing that $x_i + x_j \geq 1$ for each couple of nodes i and j:

$$y_2 = \sum_{i,j\in E}\left(1 - x_i - x_j + x_i x_j\right). \tag{29}$$

Therefore, the final objective function can be written as follows:

$$f_{\text{vertex}}(\mathbf{x}) = \sum_{i\in V} x_i + \lambda \sum_{i,j\in E}\left(1 - x_i - x_j + x_i x_j\right). \tag{30}$$

An example of a minimum vertex cover problem energy profile is reported in Figure 6g.

h: *Number partitioning*

The goal of **number partitioning** [44] optimization is to separate m positive integers belonging to a set $S = \{s_1, s_2, ..., s_m\}$, into two subsets S_1 and S_2, having an equal sum of their constituting numbers. An example is shown in Figure 5h. A binary variable x_i for **each number in the initial set** S is required for obtaining the QUBO formulation, which assumes a value of 0 if the i^{th} number is assigned to the subset S_2 and 1 otherwise.
Therefore, the sum of the numbers in the first subset is equal to:

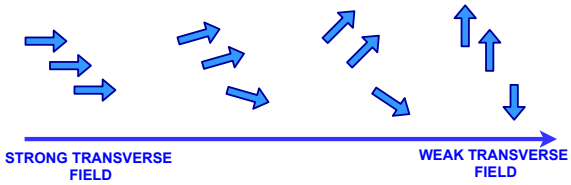
$$\text{sum}_1 = \sum_{i=1}^{m} s_i x_i\,, \tag{31}$$

while the sum of the numbers in the second one is equal to:

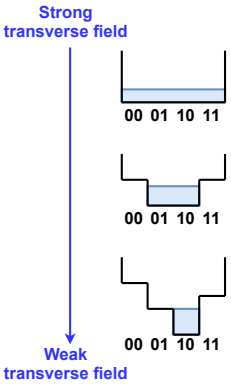
$$\text{sum}_2 = \sum_{i=1}^{m} s_i - \sum_{i=1}^{m} s_i x_i\,. \tag{32}$$

Consequently, the difference between the two sums is equal to:

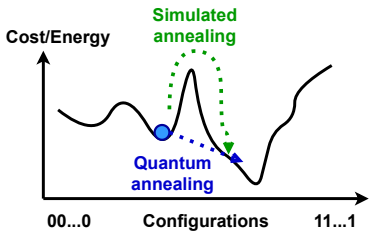
$$\text{diff} = \text{sum}_2 - \text{sum}_1 = \sum_{i=1}^{m} s_i - 2\sum_{i=1}^{m} s_i x_i\,, \tag{33}$$



(a) Spins evolution during annealing. When a strong transverse field is applied, spins are in an equiprobability state. Reducing the transverse field, spins assume a spin-down or spin-up state in order to reach the lowest energy configuration.

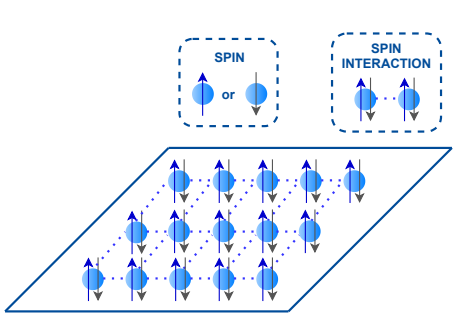


(b) Quantum annealer system evolution explained through a simple *hydraulic model*.

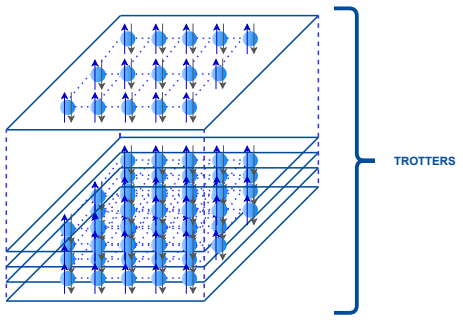


(c) Quantum tunneling *vs.* thermal climbing.

FIGURE 7: Overview of quantum annealing.



(a) A lattice of spins in an Ising model



(b) A Ising model scomposed in many Trotter slices

FIGURE 8: The classical description of the transverse field Ising model through the addition of one dimension. For simplicity all prime neighbor connections are represented.

which must be minimized to achive the target of the problem, so that the final objective function can be written as:

$$f_{\text{number}}(\mathbf{x}) = \left(\sum_{i=1}^{m} s_i - 2 \sum_{i=1}^{m} s_i x_i \right)^2 . \tag{34}$$

Similarly to the maxcut problem, Number partitioning is characterized by **symmetric energy profiles** (as shown in Figure 6h); 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 subsets are interchangeable.

i: *Linear regression*

Linear regression is another relevant problem whose applications range from scientific research [63] to business [64]. It consists in finding a linear relationship between an independent variable x, and a dependent one y. Linear regression is also employed in **supervised learning** for **modelling a target prediction value based on independent variable**.
A QUBO formulation of this problem can be obtained following the steps reported in [65], [66] in a supervised-learning-compliant notation. Given X, which is the real $N \cdot (d+1) \times N \cdot (d+1)$ matrix associated with the augmented regression training data, $Y \in \mathbb{R}^N$, which is the vector of

training labels, and $w \in \mathbb{R}^{d+1}$, which is a weights vector, linear regression corresponds to:

$$\min_{w\in\mathbb{R}^{d+1}} \quad E(w) = \|Xw - Y\|^2 , \tag{35}$$

where $E(w)$ is the Euclidean error function.
In order to obtain the QUBO formulation, regression must be rewritten as:

$$\min_{w\in\mathbb{R}^{d+1}} \quad E(w) = w^{\mathrm{T}}X^{\mathrm{T}}Xw - 2w^{\mathrm{T}}X^{\mathrm{T}}Y + Y^{\mathrm{T}}Y . \tag{36}$$

Moreover, two vectors must be defined to obtain a binary representation of weights w_i of vector w. The first one is a K-dimensional precision vector $P = [p_1, p_2, ..., p_k]^{\mathrm{T}}$ of sorted powers of 2, the other is a K-dimensional vector $\hat{w}_i^{\mathrm{T}}$ with binary coefficients such that $\hat{w}_i^{\mathrm{T}}P = w_i$. At this point, defining the binary vector $\hat{w} \in \mathbb{B}^{K(d+1)}$ as:

$$\hat{w} = [\hat{w}_{11}...\hat{w}_{1K}\hat{w}_{21}...\hat{w}_{2K}...\hat{w}_{(d+1)1}...\hat{w}_{(d+1)K}]^{\mathrm{T}} , \tag{37}$$

with all the binary variables required for representing the $d+1$ weights on K bits, and a precision matrix P as:

$$P = I_{d+1} \otimes P^{\mathrm{T}} , \tag{38}$$

where I_{d+1} is a $(d+1)$-dimensional identity matrix and $\otimes$ is the Kronecker product, the original weight vector can be binary-number approximated as:

$$w = P\hat{w} . \tag{39}$$

Substituting Equation 39 in 36, the problem is written in an equivalent QUBO form:

$$\begin{aligned}\min_{w\in\mathbb{B}^{(d+1)K}} \quad E(\hat{w}) =& \hat{w}^{\mathrm{T}}P^{\mathrm{T}}X^{\mathrm{T}}XP\hat{w} - 2\hat{w}^{\mathrm{T}}P^{\mathrm{T}}X^{\mathrm{T}}Y \\ &+ Y^{\mathrm{T}}Y ,\end{aligned} \tag{40}$$

where $Y^{\mathrm{T}}Y$ can be neglected because it introduces a scalar constant.
An example of a linear regression energy profile is reported in Figure 6i.

B. SIMULATED QUANTUM ANNEALING

Simulated Quantum Annealing (**SQA**) algorithm is a heuristic method which permits to solve combinatorial optimization problems on digital computers by emulating the exploration principles of a **quantum annealer** [67]–[77]. In particular, the SQA tries to mimic the **quantum tunnelling** effect (Figure 7c) and the **superposition principle** on classical computers by exploiting a **path integral quantum Monte Carlo simulation** (**PI**). In this way, this optimizer has the **potential** to find the global minima of an objective function faster than **simulated annealing** (**SA**) [42] and allows solving larger-problem than current **quantum annealers**, which are limited in terms of number of available qubits, have significant connectivity limitations and are affected by phenomena affecting the reliability of the obtained results.
The SQA algorithm emulates the behaviour of a quantum annealer by computing the adiabatic evolution of the Hamiltonian of **transverse-field Ising model**, expressed as:

$$\begin{aligned}H(t) &= \sum_{ij} J_{ij}\sigma_i^z\sigma_j^z + h\sum_i \sigma_i^z + \Gamma(t)\sum_i \sigma_i^x = \\ &= H_0 + \Gamma(t)\sum_i \sigma_i^x ,\end{aligned} \tag{41}$$

where H_0 is the standard Ising model on which the problem is mapped, $\Gamma(t)$ is a time-dependent transverse field which causes quantum tunneling between system eigenstates and has a similar role of temperature in simulated annealing, σ_i^x and σ_i^z are the **Pauli matrices** associated, respectively, with x and z components of the i^{th} Ising Hamiltonian spin:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} , \tag{42}$$

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} . \tag{43}$$

A qualitative idea of the quantum annealer (QA) system evolution can be given by comparing the evolution of the spin in Figure 7a and the **hydraulic model** in Figure 7b.
The transverse field initially creates the superposition of all possible states with equal weight (spins aligned on the x-axis). This condition corresponds to making the bottom of the tank flat; therefore, the water is uniformly distributed among the solutions. Then the decrease of the transverse field's strength allows the system's energy to gradually reproduce the problem's energy profile; this can be interpreted as a gradual deformation of the tank's bottom for describing the objective function, and the water begins to flow towards the lowest points. Suppose the evolution is sufficiently slow (**adiabatic**). In that case, **the system can follow the ground state for the entire evolution time**. **The final configuration is the optimal solution**, described in the hydraulic model as the water concentrated in the lowest point.
The advantage of QA exploration is that the probability of overcoming an energy barrier with height Δ is proportional to $e^{-\frac{\sqrt{\Delta}w}{\Gamma}}$ (where w is the width of the energy barrier and Γ the strength of the transverse field). At the same time, in the SA case, it is equal to $e^{-\frac{\Delta}{k_B T}}$ (where T is the temperature parameter and k_B is the Boltzmann constant). Consequently, quantum exploration is significantly more effective than classical one in case of problems with the energy landscape with a high amount of perturbation with many high and thin barriers ($w \ll \Delta$) [76].
In order to obtain an equivalent classical Ising model of the transverse field one, an **additional dimension** has to be added to the system. Indeed, an m-dimensional quantum space can be emulated by an $(m+1)$ classical one [78]. The original m-dimensional space is called **real space**, and the additional dimension is composed of a **set of interacting replicas** called **Trotter slices** (as shown in Figure 8b).

The Hamiltonian of the classical equivalent Ising model can be expressed by:

$$H = \sum_{k=1}^{M} \left(\sum_{ij} \frac{J_{ij}}{M} \sigma_{i,k} \sigma_{j,k} + \sum_{i} \frac{h_i}{M} \sigma_{i,k} + J^{+} \sum_{i} \sigma_{i,k} \sigma_{i,k+1} \right), \tag{44}$$

where $J^{+} = \frac{T}{2} \log\left(\cot\left(\frac{\Gamma}{M \cdot T}\right)\right)$ is the **correlation factor** (corresponding to the strength of couplings between replicas), depending on the temperature parameter T, the transverse field Γ and M, which is the number of replicas considered. T and Γ gradually decrease with the following schedules: $T = \frac{\text{MC\_step}}{(1-\frac{1}{8}) \cdot (t+1)}$ and $\Gamma = \Gamma_0 \cdot (1 - \frac{t}{\text{MC\_step}+1})$, where t is the number of the current iteration, Γ_0 the initial transverse field [68] and MC_step is the total number of **Monte Carlo steps**, corresponding to the update of all spins of all replicas.

The correlation factor is weak at the beginning to guarantee a **search radius as larger as possible** (**exploration**). Afterwards, it increases for reaching a high probability of the **convergence** of each replica to the same final configuration, which should be the optimal one (**exploitation**).

The Trotters or replicas employment allows the emulation of quantum system **states superposition** and, together with the correlation factor, mimics the **tunneling effect** through energy barriers, thus ensuring a faster algorithm convergence. A higher number of Trotter M implies a higher fidelity of the SQA in emulating the QA evolution.

From a certain point of view, Trotters can be seen as individuals of a population-based algorithm. In particular, a theoretical comparison with cooperative population-based algorithms, like ones in [13], [10], can be done. Indeed, the Trotters can be seen as individuals of a population that, in the beginning, tends to explore the solution space without considering other elements (selfish behaviour), but gradually the tendency to stick together grows (social behaviour).

The main steps of the algorithm are:

1) For each Trotter, an arbitrary solution is considered.
2) For each spin of each replica, an update is considered and accepted according to the **Metropolis-Hastings algorithm**.
3) Evaluate which replica gives the current best solution and if this solution is better than the ones obtained in the previous step.
4) Repeat from 2) decreasing Γ and T and updating J^{+} according to the explained formula until $\Gamma = 0$.

It is important to notice that steps from 2) to 4) constitute a single Monte Carlo step of the algorithm.

The evaluation and storage for each Trotter of the best current solution in each iteration (point 3) were inserted in the algorithm to improve the probability of obtaining the optimal solution.

The Metropolis-Hastings algorithm (MH) [79] is a popular Markov chain Monte Carlo mechanism for obtaining a sequence of random samples associated with a complex non-evaluable probability distribution. This allows the **simulation of random walks** in the solution space, i.e. a random exploration depending on the acceptance rate and generation of new samples from the previous one.

Its pseudocode is reported in Algorithm 1.

Algorithm 1 Metropolis-Hasting

```
Start from an arbitrary sample x_0
for i = 1 To i < UPDATE do
    generate x_i^* from x_{i-1}
    U = rand(0, 1)
    //A(x_{i-1}, x_i) is the Acceptance Rate
    //It depends on application
    if u < A(x_{i-1}, x_i) then
        //Accept the new sample
        x_i = x_i^*
    else
        //Maintain the old sample
        x_i = x_{i-1}
    end if
end for
```

In the SQA case, the MH acceptance rate of the n^{th} spin of the m^{th} Trotter is:

$$\mathcal{A} = \min\left(1, e^{\frac{-\Delta E \cdot M}{T}}\right), \tag{45}$$

where:

$$\Delta E = \frac{\Delta E_{\text{pot}}}{M} + \Delta E_{\text{kin}}\,. \tag{46}$$

Considering that M and T are always both positive, if $\Delta E \leq 0$, the exponential $e^{\frac{-\Delta E \cdot M}{T}}$ is at least equal to 1, so the spin-flip is accepted ($\mathcal{A} = 1$), otherwise a positive ΔE implies a spin-flip probability equal to $\mathcal{A} = e^{\frac{-\Delta E \cdot M}{T}}$. In SQA, the single-spin state is changed if ΔE_{pot} is lower than 0, thus implying that any energy improvement of the single replica is maintained. Indeed, ΔE_{pot} depends only on the configuration of the other Trotter's spins:

$$\Delta E_{\text{pot}} = \left(\sum_{k=0}^{N-1} \left(2 \cdot \text{J}_{n,k} \cdot \text{s\_m}_{m,n} \right) \text{h}_n \right) \cdot \left(-2 \cdot \text{s\_m}_{m,n}\right), \tag{47}$$

where J and h are the interaction matrix and external field vector, respectively, of the problem, s_m is the spins matrix, where each row is associated with the spins configuration of a given Trotter, N the number of spins for each Trotter, i.e. the number of involved binary variable and m and n are, respectively, the Trotter and spin index for identifying the spin to update, while k is an index for iterating among the other spins of the same Trotter. This ΔE contribution considers each replica alone.

On the contrary, ΔE_{kin} takes into account the correlation among neighbor Trotters:

$$\Delta E_{\text{kin}} = J^{+} \cdot (\text{s\_m}_{\text{m\_dx},n} + \text{s\_m}_{\text{m\_sx},n}) \cdot (-2 \cdot \text{s\_m}_{m,n})\,, \tag{48}$$

where J^{+} is the correlation factor and m_dx and m_sx are the index of the two neighbor Trotters of m^{th} one obtained as $m+1$ and $m-1$, respectively. This ΔE component increases the spin-flip acceptance if the new spin configuration is the

same as its neighbours to guarantee the convergence of all replicas at the same solution. The strength of this component increases during the algorithm execution as the correlation factor for giving more freedom during the exploration beginning and for gradually forcing the convergence.
Therefore, the Metropolis-Hastings method guarantees a good balance between exploration and exploitation during the algorithm evolution because the probability of accepting a degrading solution better separates a Trotter's configuration from its neighbours (getting out of the flock) decreases during the algorithm evolution.
A detailed and complete pseudo-code is reported in Algorithm 2.

Algorithm 2 Simulated Quantum Annealing

Input: **J** matrix and **h** vector
Output: Solution vector **s** and **Energy** value

Initialize:
  //Fill randomly spin configurations
  //matrix with -1 or 1
  Randomly initialize **s_m** $\in \{-1, 1\}$
  //Variables initialization
  Energies $\leftarrow 0$
  ΔE_local $\leftarrow 0$
  $\Gamma \leftarrow \Gamma_0$
  $T \leftarrow \frac{\text{MC\_step}}{1-\frac{1}{8}}$
  $J^+ = \frac{T}{2} \log\left(\cot\left(\frac{\Gamma}{M \cdot T}\right)\right)$
  //ΔE_local precomputation
  //Current energy evaluation
  for $m = 0$ **To** $M - 1$ **do**
    for $n = 0$ **To** $N - 1$ **do**
      for $k = 0$ **To** $N - 1$ **do**
        //Symmetric J matrix considered
        $\Delta\text{E\_local}_{n,m} \mathrel{+}= 2 \cdot \text{J}_{n,k} \cdot \text{s\_m}_{m,n}$
      end for
      $\Delta\text{E\_local}_{n,m} \mathrel{+}= \text{h}_n$
    end for
    $\text{Energies}_m = \text{s\_m}_m \times (\text{J} \times \text{s\_m}_m^{\text{T}})^{\text{T}} + \text{s\_m}_m \times \text{h}^{\text{T}}$
    //Save current optimal solution
    if $m, n = 0$ **or** Energy $\geq \text{Energies}_m$ **then**
      Energy $\leftarrow \text{Energies}_m$
      s $\leftarrow \text{s\_m}_m$
    end if
  end for

Monte Carlo steps:
  for $t = 0$ **To** MC_step $- 1$ **do**
    for $m = 0$ **To** $M - 1$ **do**
      //Identify Trotter neighbors
      Identify n_dx and n_sx //$m-1$ and $m+1$
      for $n = 0$ **To** $N - 1$ **do**
        //Evaluation of ΔE
        //for flipping the spin
        $\Delta\text{E\_pot} = -\Delta\text{E\_local}_{n,m} \cdot (2 \cdot \text{s\_m}_{m,n})$
        $\Delta\text{E\_kin} = J^+ \cdot (\text{s\_m}_{\text{n\_dx},n} + \text{s\_m}_{\text{n\_sx},n}) \cdot (-2 \cdot \text{s\_m}_{m,n})$
        $\Delta\text{E} = \frac{\Delta\text{E\_pot}}{M} + \Delta\text{E\_kin}$
        $r = rand(0, 1)$
        //Metropolis-Hasting condition
        if ΔE_pot < 0 **or** $r < e^{\frac{-\Delta\text{E} \cdot M}{T}}$ **then**
          //Spin-flip
          $\text{s\_m}_{m,n} = -\text{s\_m}_{m,n}$
          $\text{Energies}_m = \text{Energies}_m + \Delta\text{E\_pot}$
          //Update optimal solution
          if Energy $\geq \text{Energies}_m$ **then**
            Energy $\leftarrow \text{Energies}_m$
            s $\leftarrow \text{s\_m}_m$
          end if
          //Update ΔE_local
          for $k = 0$ **To** $N - 1$ **do**
            //Symmetric J matrix
            $\Delta\text{E\_local}_{K,m} \mathrel{+}= 4 \cdot \text{J}_{n,k} \cdot \text{s\_m}_{m,n}$
          end for
        end if
      end for
    end for
    //Update the evolution parameters
    $\Gamma = \Gamma_0 \cdot (1 - \frac{t}{\text{MC\_step}+1})$
    $T = \frac{\text{MC\_step}}{(1-\frac{1}{8}) \cdot (t+1)}$
    $J^+ = \frac{T}{2} \log\left(\cot\left(\frac{\Gamma}{M \cdot T}\right)\right)$
  end for

Return: Energy, s

Disadvantages of the SQA approach with respect to the SA one are:

- **high memory requirement** due to the employment of replicas;
- **time requirement** for each Monte Carlo step, even though, according to data dependencies, **it is possible to parallelize each iteration partially**.

The **parallelization** of this algorithm is **crucial** for avoiding significant growth with the number of Trotters and the involved variables of the time required for executing each iteration. For this reason, FPGA-based [69]–[73], [76] and GPU-based [67], [68], [77] algorithm implementations, to accelerate the SQA computation, have been proposed in the state-of-art. However, the **data dependencies** do not simplify parallelization. Indeed, considering a generic fully-connected problem, the update of spins belonging to the **same replica** has to be performed **serially** to respect the data dependencies. Moreover, the update of the i^{th} spin in the various replicas has to be performed serially. The only way to parallelize updates is to **update replicas in parallel, but each one is delayed by one relative position with respect to its neighbour**, as shown in Figure 9. In this way, the time required for the algorithm execution is:

$$t_{\text{exe\_parallel}} = t_{\text{init}} + ((M-1) \cdot t_{\text{spin}} + N \cdot t_{\text{spin}} + t_{\text{param}}) \cdot \text{MC\_step}\,, \tag{49}$$

where t_{init} is the initialization time, t_{spin} is the time required for the update of a single spin (inner loop), and t_{param} is the one required for updating the evolution parameters. On the other hand, the fully-sequential execution time would be:

$$t_{\text{exe\_serial}} = t_{\text{init}} + ((M \cdot N) \cdot t_{\text{spin}} + t_{\text{param}}) \cdot \text{MC\_step}\,. \tag{50}$$

The savings is equal to $(N \cdot (M-1) + 1 - M) \cdot t_{\text{spin}} \cdot \text{MC\_step}$, which is less than $(N \cdot (M-1)) \cdot t_{\text{spin}} \cdot \text{MC\_step}$ that could be obtained completely parallelizing the Trotters evolution, but it remains a very good result.
A possibility to increase the parallelization degree is to

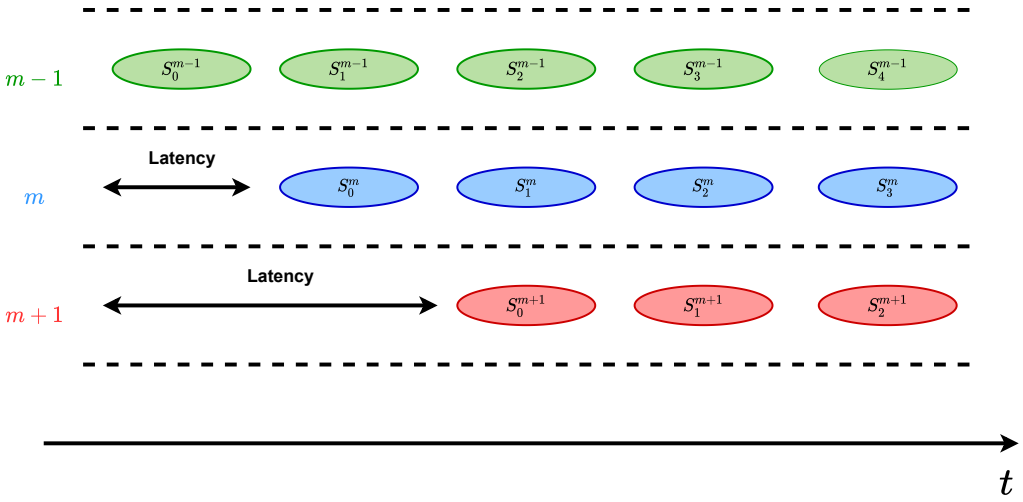


FIGURE 9: Scheduling of Monte Carlo step t, computing Trotter updates in parallel according to data dependencies.

consider a **minor embedding** [80] technique to decrease the connectivity of each spin and to identify **groups of spins that are entirely independent of the other groups**. In this way, these spin updates can be executed in parallel to the others if there are sufficient hardware resources. However, it has to be taken into account that these techniques **increase the total number of spins involved** and that their application involves an initial time overhead. Therefore it is good to use this technique only if the available hardware is sufficient.

C. PARALLEL TEMPERING

Parallel Tempering (**PT**) [81]–[83] or **replicas exchange** technique is based on the simulation of **multiple copies of the original system** of interest. Each of them works at a **different temperature**. The spins update is done according to the Metropolis-Hastings algorithm, depending on the acceptance rate of classical SA. Substantially each copy executes the Monte Carlo step in a different *instant of time* of the SA algorithm, as shown in Figure 11.

The systems with a **higher temperature** have a **more extensive search radius** in the solution space, while the **lower** ones perform **local exploration**. PT performs better than SA because **temperature swapping** between the copies can be performed in each iteration of the algorithm. The probability of **swapping the temperatures** is computed to allow copies having **poor results** (higher energy) until that moment to have a high probability of accepting a higher temperature parameter, thus **growing the search radius**. In particular, the swap probability between i^{th} and j^{th} copies is equal to:

$$P_{\text{swap}}(i,j) = \min\left[1, e^{(\frac{1}{T_i}-\frac{1}{T_j})(E_i-E_j)}\right], \qquad (51)$$

where is T_i and T_j are the temperatures of the i^{th} and j^{th} copy, respectively, and E_i and E_j are the copies energy.

This mechanism can allow copies stuck in **local optima** to be bumped out of them, thus encouraging a broader exploration of the problem space for poor-performance copies and forcing a **restricted search** for those providing **good performance** (low energy).

For summarizing, the main steps of the algorithm are:

1) For each copy, an arbitrary solution is considered and a temperature parameter is assigned.
2) Spin update for each copy is considered and accepted according to the **Metropolis-Hastings algorithm**.
3) Evaluate the current energy of each copy.
4) **Temperature swapping** according to the energy of the copies.
5) Repeat from 2).

Steps from 2) to 5) constitute a PT iteration.

The main difference with standard SA evolutions computed in parallel is that, instead of linearly decreasing the temperature parameter, this is set in each iteration for each copy according to the quality of solutions reached until that moment.

From a certain point of view, it can be seen as a population-based version of SA, which increases the exploration of the worst population elements and increases the exploitation of the best ones.

An advantage of PT is that the Monte Carlo step of each copy can be done perfectly in **parallel**, so the overhead related to the employment of many system copies can be compensated in a hardware implementation.

D. POPULATION ANNEALING

Population Annealing (**PA**) [84]–[86] is a sequential Monte Carlo method whose target is attenuating the Metropolis-Hastings algorithm's vulnerability to **rough energy landscapes** (i.e. a multimodal objective function with multiple minima) by **simulating a population of Metropolis walkers**. It is closely related to SA, but also, in this case, it involves **a population of copies**, which is resampled at each temperature step. Similarly to SA, PA **monotonically lowers the system temperature** through a sequence of temperatures from high to low. However, differently from SA, a **re-**

sampling step is performed at each temperature. Therefore, the solution quality of each copy is evaluated and, according to its quality with respect to the others and exploiting a **Poisson's probability density function**, its spins configuration is eliminated, kept or duplicated a certain number of times, in place of the spin configurations of the eliminated copies, as shown in Figure 12. The average number of copies to be kept for the i^{th} original one can be estimated as follows:

$$N(E_i) = \frac{1}{Q} \exp\left(\left(\frac{1}{T(t')} - \frac{1}{T(t'+1)}\right) E_i\right), \qquad (52)$$

where E_i is the energy of the i^{th} copy, $T(t')$ and $T(t'+1)$ are, respectively, the temperature in the current t' and in the next steps of the algorithm and Q is a **normalization factor** equal to:

$$Q = \frac{1}{M} \sum_{i=0}^{M-1} \exp\left(\left(\frac{1}{T(t')} - \frac{1}{T(t'+1)}\right) E_i\right), \qquad (53)$$

where M is the number of copies in the population.
The re-sampling mechanism consolidates the search efforts around the **most promising regions**. Indeed, each copy is free to explore its search region, but whenever a copy finds a favourable one, the rest of the population is moved toward it. As it is possible to understand from the reported formula, the re-sampling mechanism is enforced during the algorithm evolution. Initially, it is weak for favouring a broad exploration of the solution space. At the same time, with low-temperature values, the combination of Metropolis criteria and a strong re-sampling, a local search focused on the region of the competitive state is performed, and the search in the poor regions is aborted.
In this sense, the PA algorithm shares the basic principles of **evolutionary population-based approaches** [87] [17] [88]. Indeed, the algorithm evolution starts with a generation of copies (individuals) and a sort of selection of the best elements (configurations) associated with the lowest values for creating a new generation (exploitation). The balance with exploration is guaranteed by the employment of Poisson's probability density function, which permits, with a low probability, the selection of the temporary worst solutions.
For summarizing, the main steps of the algorithm are:

1) For each copy, an arbitrary solution is considered.
2) Spin update for each copy is considered and accepted according to the **Metropolis-Hastings algorithm**.
3) Evaluate the current energy of each copy.
4) **re-sampling** copies according to their energy.
5) Repeat from 2) decreasing the temperature parameter T.

Steps from 2) to 5) constitute a PA iteration.
Similar to PT, a further advantage of this approach is that the Monte Carlo step of each copy can be done entirely in **parallel**, so the overhead related to the employment of many system copies can be compensated in a hardware implementation. The resource requirement for parallelizing the computation makes this approach more suitable for challenging moderately-sized problems. However, it could have some difficulties in exploring regions with high peaks [89].

III. HYBRID QUANTUM-CLASSICAL ALGORITHMS

It is possible to prove that the effectiveness of an optimization algorithm on a specific problem is strongly related to the characteristic of its energy profile [43].
For example, SA and other **local-search-based** approaches can easily overcome **wide** and **smooth barriers**, but they cannot effectively overcome **high** and **narrow barriers**.
On the contrary, QA performs well with problems whose energy profiles are characterized by **high** and **narrow peaks**, thanks to the exploitation of quantum tunnelling. However, it is expected to be ineffective in **vast** and **flat regions** [90] [91]. For these reasons, the exploration performed by QA can be defined as **global**.
Unfortunately, the energy profiles associated with real-world problems are usually **heterogeneous**, as reported in Figure 10. Therefore, the probability of success of each solver depends on the size of the energy profile region compatible with its exploration mechanism. In the case of a heterogeneous energy profile, a significant advantage can be obtained by employing a solver which can explore the wide and smooth region with a local approach and the rough one with QA.
From these observations, the interest in **hybrid solvers**, which can effectively alternate **local** and **global searches**, arose and some proposals were developed [92]–[94].
N. Chancellor proposed in [43] techniques to perform a local search rather than a global one in a QA to obtain a hybrid solver, which maintains both the advantages of the QA and SA. In particular, he proposed the application to QA of some techniques born to improve classical SA to achieve better QA performance for heterogeneous energy profiles. For example, Parallel Tempering (PT) and Population Annealing (PA), presented in Paragraphs II-C and II-D, respectively, were considered.
He demonstrates how sequential calls to QA can be exploited to obtain analogues of PT and PA, employing quantum search as a subroutine. Unfortunately, the QA is essentially different from SA because, due to the no-cloning theorem of quantum mechanics, it is impossible to determine the system's intermediate state; consequently, they cannot be arbitrarily manipulated to build a better algorithm. Therefore, obtaining the quantum version requires a **subroutine** similar to QA, performing a **local search in a controllable-size region around the initial state**. Furthermore, since the input and output of the subroutine are entirely classical, the no-cloning theorem can be considered non-critical, and this local quantum subroutine can be combined with a quantum or classical search. Moreover, N. Chancellor identifies an **effective temperature parameter** to describe parallel-tempering-like and population-annealing-like mechanisms, which can be derived from the following Hamiltonian, describing the

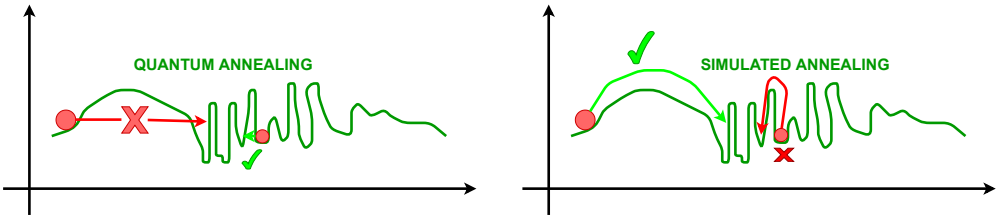


FIGURE 10: Exploration capabilities and limits of SA and QA with an **heterogeneous** energy profile. The wide energy barrier can be overcome by SA, exploiting thermal fluctuations but cannot be tunnelled by QA. On the other hand, QA can efficiently explore the rough part of the energy landscape, while SA cannot overcome this region's high and narrow peaks.

adiabatic evolution of a single qubit of a quantum annealer:

$$H_1(t) = -A(t)\sigma^x + B(t)\sigma^z\,. \tag{54}$$

$A(t)$ is the transverse field, $B(t)$ is a longitudinal field, which permits to gradual apply the problem weight (thus implementing the adiabaticity of the QA evolution), t is the time-instant in the annealing schedule, and σ^x and σ_z are the Pauli matrices. The two fields have a complementary behaviour with time to ensure the superposition of the initial states ($A(t)$) and the final convergence to the optimal solution of the analyzed problem ($B(t)$).
Eigenstates and eigenvalues of the Hamiltonian can be analytically evaluated by exploiting diagonalization and the ratio among the eigenvalues of the two basis states can be produced:

$$\frac{\psi(1)}{\psi(2)} = \frac{\sqrt{A(t)^2 + B(t)^2}}{A(t)} + \frac{B(t)}{A(t)}\,. \tag{55}$$

Finally, the **effective temperature** can be derived by comparing the quantum probability distribution to a Boltzmann distribution on only the longitudinal part of the reported Hamiltonian

$$T_{\text{eff}}(t) \triangleq 2\left[\ln\left(\left|\frac{\sqrt{A(t)^2 + B(t)^2}}{A(t)} + \frac{B(t)}{A(t)}\right|^2\right)\right]^{-1}. \tag{56}$$

This work proposes new hybrid algorithms obtained by combining the SQA, PT and PA starting from Chancellor's intuition and observations. In particular, Equation 56 was adapted to the system considered in SQA, described in Paragraph II-B, recognizing that $A(t)$ plays essentially the role of $-\Gamma(t)$ and $B(t)$ is fixed at one:

$$T_{\text{eff}}(t) = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma(t)^2 + 1}}{\Gamma(t)} + \frac{1}{\Gamma(t)}\right|^2\right)\right]^{-1}, \tag{57}$$

in which the minus signs in the formula are omitted due to the presence of the square module. Moreover, the **system copies** of PT and PA are interpreted as **copies of the correlated Trotters system** described in Figure 8b.
Finally, this article proposes four new algorithms obtained by exploiting the presented principles: **Simulated Quantum Parallel Tempering** (**SQPT**), **Simulated Quantum Population Annealing** (**SQPA**), **Simulated Quantum Parallel Tempering - Population Annealing version 1** (**SQPTPA1**) and **Simulated Quantum Parallel Tempering - Population Annealing version 2** (**SQPTPA2**). They are analyzed and described deeply in the following Paragraphs.

A. SIMULATED QUANTUM PARALLEL TEMPERING

Simulated Quantum Parallel Tempering (**SQPT**) was proposed to join the advantages of SQA and PT algorithms. As mentioned, it was substantially implemented by considering **each system copy of the standard PT composed of SQA Trotters** and its spin-update was performed according to SQA conditions. Moreover, the time instants, i.e. of the transverse field $\Gamma(t)$ and correlation factor $J^+(t)$, assigned to each system copy are different and they are swapped among the system copy after each iteration according to the system energy and the effective temperature of Equation 57. In particular, the swap probability is equal to:

$$P_{\text{swap}}(i,j) = \min\left[1, e^{(\frac{1}{T_{\text{eff}_i}} - \frac{1}{T_{\text{eff}_j}})(E_i - E_j)}\right], \tag{58}$$

where E_i and E_j are the energies of the best configuration of the i^{th} and j^{th} systems, respectively.
A graphical idea of the algorithm behavior is provided in Figure 11, which shows the evolution of the system. This is composed of a set of copies, composed in turn of Trotters slices reported in Figure 8b.
The main evolution steps are the following:

1) For each Trotter of each system, an arbitrary solution is considered.
2) At a given time instant t, a transverse field $\Gamma(t)$, a correlation factor $J^+(t)$ and a temperature parameter $T(t)$ are assigned to each system copy from a uniformly distributed list in range 0 to $\text{MC}_{\text{step}} - 1$.
3) For each spin of each replica of each system copy, an update is considered and accepted according to the **Metropolis-Hastings algorithm**.
4) Evaluate which replica gives the current best solution and if this solution is better than the ones obtained in the previous step.
5) Save the best current solution of each system copy.
6) Evaluate, for each couple of copies of the system, the time instants swap according to the MH algorithm with an acceptance rate equal to Equation 58.
7) Repeat from 2) for MC_{step} number of times.

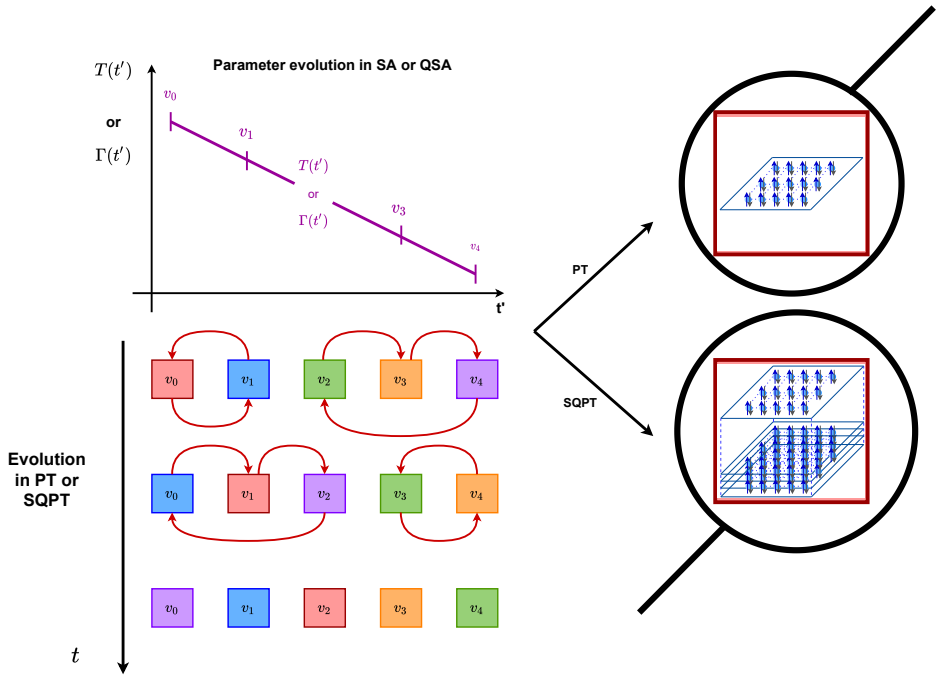


FIGURE 11: Parallel Tempering and Simulated Quantum Parallel Tempering evolution.

In this case, steps 2) to 7) constitute a single Monte Carlo step of the algorithm.

Analogously to SQA, the flip of the i^{th} spin of the m^{th} Trotter of the s^{th} system copy is accepted if ΔE is non-positive or if ΔE_{pot} is lower than 0 or with a probability equal to $e^{\frac{-\Delta E \cdot M}{T(s)}}$. The only difference is that the correlation factor J^+ considered in the ΔE_{kin} computation and the temperature T are not functions of the current iteration but of the considered system copy. Indeed, the algorithm parameters Γ, T and J^+ do not have a monotonic evolution with time, but their values change with time depending on the system copies energies according to the swap mechanism.

A detailed and complete pseudo-code is reported in Algorithm 3.

Algorithm 3 Simulated Quantum Parallel Tempering

Input: **J** matrix and **h** vector
Output: Solution vector **sol** and **Energy** value

Initialize:
//Fill randomly spin configurations
//tensor $N \times M \times SYS$ with -1 or 1
//where SYS is the number of system
//copies
Randomly initialize **s_m** $\in \{-1, 1\}$
//Variables initialization
Energies $\leftarrow 0$
ΔE_local $\leftarrow 0$
//Vector of systems optimal energy
$E_{\text{sys\_opt}} \leftarrow 0$
//Precomputation of the involved Γ, T
//and J^+ parameters
$\Gamma_{\text{list}} \leftarrow \text{linspace}\left(\Gamma_0, \Gamma_0\left(1 - \frac{\text{MC}_{\text{step}}}{\text{MC}_{\text{step}}}\right)\right), SYS\right)$
$T_{\text{list}} \leftarrow \text{linspace}\left(\frac{\text{MC\_step}}{1-\frac{1}{8}}, \frac{\text{MC\_step}}{(1-\frac{1}{8})\cdot(\text{MC\_step})}, SYS\right)$
$J^+_{\text{list}} \leftarrow \frac{T_{\text{list}}}{2} \log\left(\cot\left(\frac{\Gamma_{\text{list}}}{M \cdot T_{\text{list}}}\right)\right)$
//Systems time instant initialization
$s \leftarrow \text{linspace}(0, SYS - 1, SYS)$
//ΔE_local precomputation
//Current energy evaluation
for $sys = 0$ **To** $SYS - 1$ **do**
  for $m = 0$ **To** $M - 1$ **do**
    for $n = 0$ **To** $N - 1$ **do**
      for $k = 0$ **To** $N - 1$ **do**
        //Symmetric J matrix
        $\Delta\text{E\_local}_{n,m,sys}$ += $2 \cdot \text{J}_{n,k} \cdot \text{s\_m}_{m,n,sys}$
      end for
      $\Delta\text{E\_local}_{n,m,sys}$ += h_n
    end for
    $\text{Energies}_{m,sys} = \text{s\_m}_{m,sys} \times (\text{J} \times \text{s\_m}^{\text{T}}_{m,sys})^{\text{T}}$
      $+\text{s\_m}_{m,sys} \times \text{h}^{\text{T}}$
    //Save current optimal solution
    if $m, n = 0$ **or** Energy $\geq \text{Energies}_{m,sys}$ **then**
      Energy $\leftarrow \text{Energies}_{m,sys}$
      sol $\leftarrow \text{s\_m}_m$
    end if
    //Save current optimal solution
    //system energy
    if $m = 0$ **or** $E_{\text{sys\_opt}_{sys}} \geq \text{Energies}_{m,sys}$ **then**
      $E_{\text{sys\_opt}_{sys}} \leftarrow \text{Energies}_{m,sys}$
    end if
  end for
end for

Monte Carlo steps:
for $t = 0$ **To** MC_step $- 1$ **do**
  for $sys = 0$ **To** $SYS - 1$ **do**
    for $m = 0$ **To** $M - 1$ **do**
      //Identify Trotter neighbors
      Identify n_dx and n_sx //$m - 1$ and $m + 1$

for $n = 0$ **To** $N-1$ **do**
//Evaluation of ΔE
//for flipping the spin
ΔE_pot $= -\Delta$E_local$_{n,m,sys}$ $\cdot(2 \cdot$ s_m$_{m,n,sys})$
ΔE_kin $= J^{+}_{\text{list}}(s_{sys}) \cdot (s_m_{\text{n_dx},n,sys}+$ +s_m$_{\text{n_sx},n,sys}) \cdot (-2 \cdot$ s_m$_{m,n,sys})$
$\Delta E = \frac{\Delta \text{E_pot}}{M} + \Delta$E_kin
$r = rand(0,1)$
//Metropolis-Hasting condition
if ΔE_pot < 0 **or** $r < e^{\frac{-\Delta E \cdot M}{T_{\text{list}}(s_{sys})}}$ **then**
//Spin-flip
s_m$_{m,n,sys} = -$s_m$_{m,n,sys}$
Energies$_{m,sys}$ = Energies$_{m,sys}$ + ΔE_pot
//Update optimal solution
if Energy $\geq$ Energies$_{m,sys}$ **then**
Energy $\leftarrow$ Energies$_{m,sys}$
sol $\leftarrow$ s_m$_{m,sys}$
end if
//Update ΔE_local
if $m = 0$ **or** $E_{\text{sys\_opt}_{sys}} \geq$ Energies$_{m,sys}$ **then**
$E_{\text{sys\_opt}_{sys}} \leftarrow$ Energies$_{m,sys}$
end if
for $k = 0$ **To** $N-1$ **do**
//Symmetric J matrix
ΔE_local$_{k,m,sys}$ += $4 \cdot$ J$_{n,k} \cdot$ s_m$_{m,n,sys}$
end for
end if
end for
end for
end for
//Update the evolution parameters
for $sys_1 = 0$ **To** $SYS-1$ **do**
for $sys_2 = 0$ **To** $SYS-1$ **do**
Compute $T_{\text{eff}_{sys_1}}$
Compute $T_{\text{eff}_{sys_2}}$
//Metropolis-Hasting condition
if $r < e^{(\frac{1}{T_{\text{eff}_{sys_1}}} - \frac{1}{T_{\text{eff}_{sys_2}}})(E_{\text{sys\_opt}_{sys_1}} - E_{\text{sys\_opt}_{sys_2}})}$ **then**
//Time instants swap
$s_{sys_1} \leftrightarrow s_{sys_1}$
end if
end for
end for
end for

Return: Energy, sol

A further advantage of the presented approach is that the spins update of each system copy can be done completely in parallel to the others without delays. In this way, the time required for the algorithm execution is:

$$t_{\text{exe\_parallel}} = t_{\text{init}} + ((M-1) \cdot t_{\text{spin}} + N \cdot t_{\text{spin}} + t_{\text{swap}}) \cdot \text{MC\_step}\,, \tag{59}$$

where t_{init} is the initialization time, t_{spin} is the time required for the update of a single spin (inner loop) and t_{swap} is the one required for swapping the system copies time instant, instead of:

$$t_{\text{exe\_serial}} = t_{\text{init}} + ((M \cdot N \cdot SYS) \cdot t_{\text{spin}} + t_{\text{swap}}) \cdot \text{MC\_step}\,. \tag{60}$$

The savings is equal to $(M \cdot N \cdot SYS - M + 1 - N) \cdot t_{\text{spin}} \cdot$ MC_step. The comparison of $t_{\text{exe\_parallel}}$ with one of SQA in the same condition, i.e. with the same total number of Ising copies ($M_{\text{SQA}} = M_{\text{SQPT}} \cdot SYS_{\text{SQPT}}$), for spins update, the saving is equal to $(M_{\text{SQPT}} \cdot (SYS_{\text{SQPT}} - 1) \cdot t_{\text{spin}}) \cdot$ MC_step. In a hardware implementation, this time could compensate for the overhead related to the swapping time, which will be higher than the parameter update time of SQA, especially if fast solutions like look-up tables are considered for computing the swap probabilities and the effective temperature. Therefore, in this work, the iterations of the two algorithms are considered comparable.

B. SIMULATED QUANTUM POPULATION ANNEALING

Simulated Quantum Population Annealing (**SQPA**) was proposed to join the advantages of SQA and PA algorithms. As mentioned, it was implemented substantially by considering **each system copy of the standard PA composed of SQA Trotters** and its spin-update was performed according to SQA conditions. Moreover, the evolution of the parameters is the same as SQA. However, after each Monte Carlo step, the involved system copies are **resampled**, i.e. the best solution of each system is evaluated and, according to its quality with respect to the others and **Poisson's probability density function**, is eliminated, kept or duplicated a certain number of times in place of the eliminated ones, as shown in Figure 12. The average number of system copies to be kept for the i^{th} original one can be estimated as follows:

$$N(E_i) = \frac{1}{Q} \exp\left(\left(\frac{1}{T_{\text{eff}}(t)} - \frac{1}{T_{\text{eff}}(t+1)}\right) E_i\right), \tag{61}$$

where E_i is the best energy of the i^{th} system copy, $T_{\text{eff}}(t)$ and $T_{\text{eff}}(t+1)$ are, respectively, the effective temperatures, evaluated by exploiting Equation 57, in the current and the next steps of the algorithm and Q is a **normalization factor** equal to:

$$Q = \frac{1}{SYS} \sum_{i=0}^{SYS-1} \exp\left(\left(\frac{1}{T_{\text{eff}}(t')} - \frac{1}{T_{\text{eff}}(t'+1)}\right) E_i\right), \tag{62}$$

where SYS is the number of system copies in the population. In order to be precise, the probability of obtaining a number of system copies to be kept is equal to N_{copies} is equal to:

$$P_{\text{poisson}}(N_{\text{copies}} = k) = \frac{\lambda^k e^{-\lambda}}{k!}\,, \tag{63}$$

where $\lambda = N(E_i)$ is the expected value of the Poisson distribution.

The main algorithm evolution steps are the following:

1) For each Trotter of each system, an arbitrary solution is considered.
2) For each spin of each replica of each system copy, an update is considered and accepted according to the **Metropolis-Hastings algorithm**.
3) Evaluate which replica gives the current best solution and if this solution is better than the ones obtained in the previous step.

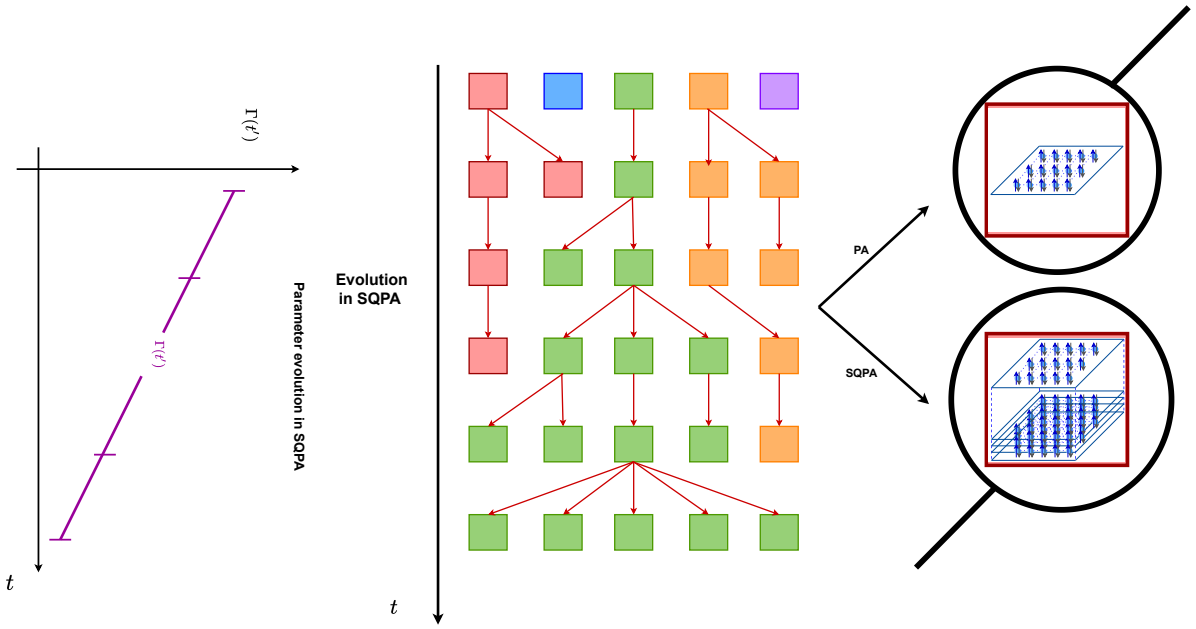


FIGURE 12: Population Annealing and Simulated Quantum Population Annealing evolution.

4) Save the best current solution of each system copy.
5) **re-sampling** system copies according to their energies.
6) Repeat from 2), updating $\Gamma(t)$, $J^+(t)$ and $T(t)$ parameters

In this case, steps 2) to 6) constitute a single Monte Carlo step of the algorithm.

The i^{th} spin of the m^{th} Trotter of the s^{th} system copy flip, analogously to SQA, is accepted if ΔE is non-positive or if ΔE_{pot} is lower than 0 or with a probability equal to $e^{\frac{-\Delta E \cdot M}{T(s)}}$. The parameters evolution is monotonic with time as in SQA. The copies re-sampling is done by computing Q as a function of the energy of the best configuration of each copy, computing $N(E_i)$ for the i^{th} system copy and sampling the number of copy to be kept from a Poisson distribution centered in $N(E_i)$. A detailed and complete pseudo-code is reported in Algorithm 4.

Algorithm 4 Simulated Quantum Population Annealing

Input: **J** matrix and **h** vector
Output: Solution vector **sol** and **Energy** value

Initialize:
//Fill randomly spin configurations
//tensor $N \times M \times SYS$ with -1 or 1
//where SYS is the number of system
//copies
Randomly initialize **s_m** $\in \{-1, 1\}$
//Variables initialization
Energies $\leftarrow 0$
ΔE_local $\leftarrow 0$
//Vector of systems optimal energy
$E_{\text{sys\_opt}} \leftarrow 0$
//Precomputation of the involved Γ, T
//and J^+ parameters
$\Gamma \leftarrow \Gamma_0$
$T \leftarrow \frac{\text{MC\_step}}{1-\frac{1}{8}}$
$J^+ = \frac{T}{2} \log\left(\cot\left(\frac{\Gamma}{M \cdot T}\right)\right)$
//Systems time instant initialization
//ΔE_local precomputation
//Current energy evaluation
for $sys = 0$ **To** $SYS - 1$ **do**
  for $m = 0$ **To** $M - 1$ **do**
    for $n = 0$ **To** $N - 1$ **do**
      for $k = 0$ **To** $N - 1$ **do**
        //Symmetric J matrix
        $\Delta\text{E\_local}_{n,m,sys} \mathrel{+}= 2 \cdot \text{J}_{n,k} \cdot \text{s\_m}_{m,n,sys}$
      end for
      $\Delta\text{E\_local}_{n,m,sys} \mathrel{+}= \text{h}_n$
    end for
    $\text{Energies}_{m,sys} = \text{s\_m}_{m,sys} \times (\text{J} \times \text{s\_m}^{\text{T}}_{m,sys})^{\text{T}}$
      $+\text{s\_m}_{m,sys} \times \text{h}^{\text{T}}$
    //Save current optimal solution
    if $m, n = 0$ **or** Energy $\geq \text{Energies}_{m,sys}$ **then**
      Energy $\leftarrow \text{Energies}_{m,sys}$
      sol $\leftarrow \text{s\_m}_m$
    end if
    //Save current optimal solution
    //system energy
    if $m = 0$ **or** $E_{\text{sys\_opt}_{sys}} \geq \text{Energies}_{m,sys}$ **then**
      $E_{\text{sys\_opt}_{sys}} \leftarrow \text{Energies}_{m,sys}$
    end if
  end for
end for

Monte Carlo steps:
for $t = 0$ **To** MC_step $- 1$ **do**
  for $sys = 0$ **To** $SYS - 1$ **do**
    for $m = 0$ **To** $M - 1$ **do**
      //Identify Trotter neighbors
      Identify n_dx and n_sx //$m - 1$ and $m + 1$
      for $n = 0$ **To** $N - 1$ **do**
        //Evaluation of ΔE
        //for flipping the spin
        $\Delta\text{E\_pot} = -\Delta\text{E\_local}_{n,m,sys}$

$\cdot(2 \cdot \text{s\_m}_{m,n,sys})$
$\Delta\text{E\_kin} = J^{+}_{\text{list}}(s_{sys}) \cdot (\text{s\_m}_{\text{n\_dx},n,sys} +$
$+\text{s\_m}_{\text{n\_sx},n,sys}) \cdot (-2 \cdot \text{s\_m}_{m,n,sys})$
$\Delta\text{E} = \frac{\Delta\text{E\_pot}}{M} + \Delta\text{E\_kin}$
$r = rand(0,1)$
//Metropolis-Hasting condition
if $\Delta\text{E\_pot} < 0$ **or** $r < e^{\frac{-\Delta\text{E}\cdot M}{T_{\text{list}}(s_{sys})}}$ **then**
//Spin-flip
$\text{s\_m}_{m,n,sys} = -\text{s\_m}_{m,n,sys}$
$\text{Energies}_{m,sys} = \text{Energies}_{m,sys} + \Delta\text{E\_pot}$
//Update optimal solution
if $\text{Energy} \geq \text{Energies}_{m,sys}$ **then**
$\text{Energy} \leftarrow \text{Energies}_{m,sys}$
$\text{sol} \leftarrow \text{s\_m}_{m,sys}$
end if
//Update ΔE_local
if $m = 0$ **or** $E_{\text{sys\_opt}_{sys}} \geq \text{Energies}_{m,sys}$ **then**
$E_{\text{sys\_opt}_{sys}} \leftarrow \text{Energies}_{m,sys}$
end if
for $k = 0$ **To** $N-1$ **do**
//Symmetric J matrix
$\Delta\text{E\_local}_{k,m,sys} \mathrel{+}= 4 \cdot \text{J}_{n,k} \cdot$
$\text{s\_m}_{m,n,sys}$
end for
end if
end for
end for
end for
//re-sampling
$Q \leftarrow 0$
$\Gamma_{\text{next}} = \Gamma_0 \cdot (1 - \frac{t+1}{\text{MC\_step}+1})$
for $sys0$ **To** $SYS-1$ **do**
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma^2+1}}{\Gamma} + \frac{1}{\Gamma}\right|^2\right)\right]^{-1}$
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma^2_{\text{next}}+1}}{\Gamma_{\text{next}}} + \frac{1}{\Gamma_{\text{next}}}\right|^2\right)\right]^{-1}$
$Q \mathrel{+}= \exp\left(\left(\frac{1}{T_{\text{eff1}}} - \frac{1}{T_{\text{eff2}}}\right) E_{\text{sys\_opt}_{sys}}\right)$
end for
$Q = \frac{Q}{SYS}$
//declare temporary tensor $N \times M \times SYS$
s_m_{temp}
$E_{\text{sys\_opt}_{\text{temp}}}$
$\Delta\text{E\_local}_{\text{temp}}$
//Variable for counting copies
$c \leftarrow 0$
for $sys0$ **To** $SYS-1$ **do**
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma^2+1}}{\Gamma} + \frac{1}{\Gamma}\right|^2\right)\right]^{-1}$
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma^2_{\text{next}}+1}}{\Gamma_{\text{next}}} + \frac{1}{\Gamma_{\text{next}}}\right|^2\right)\right]^{-1}$
$N_{\text{mean}} = \frac{1}{Q}\exp\left(\left(\frac{1}{T_{\text{eff1}}} - \frac{1}{T_{\text{eff1}}}\right) E_{\text{sys\_opt}_{sys}}\right)$
$R = \text{Poisson}(N)$
for $r = 0$ **To** $R-1$ **do**
//For maintaining constant the
//number of system
if $c < SYS$ **then**
$\text{s\_m}_{\text{temp}_c} = \text{s\_m}_{sys}$
$E_{\text{sys\_opt}_{\text{temp}_c}} = E_{\text{sys\_opt}_{sys}}$
$\Delta\text{E\_local}_{\text{temp}_c} = \Delta\text{E\_local}_{sys}$
$c \mathrel{+}= 1$
end if
end for
end for
//For maintaining constant the number
//of system
while $c < SYS$ **do**
$\text{s\_m}_{\text{temp}_c} = \text{s\_m}_{SYS-1}$
$E_{\text{sys\_opt}_{\text{temp}_c}} = E_{\text{sys\_opt}_{SYS-1}}$
$\Delta\text{E\_local}_{\text{temp}_c} = \Delta\text{E\_local}_{\text{temp}}$
$c \mathrel{+}= 1$
end while
$\text{s\_m} = \text{s\_m}_{\text{temp}}$
$E_{\text{sys\_opt}} = E_{\text{sys\_opt}_{\text{temp}}}$
$\Delta\text{E\_local} = \Delta\text{E\_local}_{\text{temp}}$
//Update the evolution parameters
$\Gamma = \Gamma_0 \cdot (1 - \frac{t}{\text{MC\_step}+1})$
$T = \frac{\text{MC\_step}}{(1-\frac{1}{8})\cdot(t+1)}$
$J^{+} = \frac{T}{2}\log\left(\cot\left(\frac{\Gamma}{M \cdot T}\right)\right)$
end for

Return: Energy, sol

A further advantage of the presented approach is that the spins update of each system copy can be done completely in parallel to the other without delays. In this way, the time required for the algorithm execution is:

$$t_{\text{exe\_parallel}} = t_{\text{init}} + ((M-1)\cdot t_{\text{spin}} + N \cdot t_{\text{spin}} + t_{\text{re-sampling}}) \cdot \text{MC\_step}\,, \tag{64}$$

where t_{init} is the initialization time, t_{spin} is the time required for the update of a single spin (inner loop) and $t_{\text{re-sampling}}$ is the one required for re-sampling system copies, instead of:

$$t_{\text{exe\_serial}} = t_{\text{init}} + ((M \cdot N \cdot SYS) \cdot t_{\text{spin}} + t_{\text{re-sampling}}) \cdot \text{MC\_step}\,. \tag{65}$$

The savings is equal to $(M \cdot N \cdot SYS - M + 1 - N) \cdot t_{\text{spin}} \cdot$ MC_step. By comparing $t_{\text{exe\_parallel}}$ with one of SQA in the same condition, i.e. with the same total number of Ising copy ($M_{\text{SQA}} = M_{\text{SQPA}} \cdot SYS_{\text{SQPT}}$), for spins update, the saving is equal to $(M_{\text{SQPA}} \cdot (SYS_{\text{SQPT}} - 1) \cdot t_{\text{spin}}) \cdot$ MC_step. From the spins update time point of view, the SQPT and the SQPA are perfectly equivalent. Indeed, in both cases, it is possible to perfectly parallelize the system copies evaluations. In a hardware implementation, this time could compensate for the overhead related to the re-sampling time, which will be higher than the parameter update time of SQA, especially if fast solutions will be found for obtaining the random number of copies to be kept for each according to the Poisson distribution. Therefore, in this work, the iterations of the two algorithms are considered comparable and comparable with the ones of the SQPT.

C. SIMULATED QUANTUM PARALLEL TEMPERING - POPULATION ANNEALING VERSION 1

Simulated Quantum Parallel Tempering - Population Annealing version 1 (**SQPTPA1**) was proposed to exploit both the advantages of SQPT and SQPA algorithms to obtain a solver as complete as possible. It was obtained substantially by running at the same time both the SQPT and the SQPA. In particular, a system of SYS_{temp} plus SYS_{pop} copies composed of SQA Trotter systems was considered. The SYS_{temp}

copies evolve according to the SQPT algorithm, while the SYS_{pop} copies follow the SQPA one. The system copies related to SQPT and ones related to SQPA are, so it is equivalent to running the two algorithms in parallel and choosing the optimal solution with a majority voting mechanism. This should allow an increase in the capability of solving problems of different types.

The main algorithm evolution steps are the following:

1) For each Trotter of each system, an arbitrary solution is considered.
2) At a given time instant t, a transverse field $\Gamma(t)$, a correlation factor $J^+(t)$ and a temperature parameter $T(t)$ are assigned to each SQPT system copy from a uniformly distributed list in range 0 to $\text{MC}_{\text{step}} - 1$.
3) For each spin of each replica of each system copy, an update is considered and accepted according to the MH algorithm.
4) Evaluate which replica gives the current best solution and if this solution is better than the ones obtained in the previous step.
5) Save the best current solution of each system copy.
6) Evaluate for each couple of SQPT system copies the time instants swap according to the MH algorithm with an acceptance rate equal to Equation 58.
7) **re-sampling** the SQPA system copies according to their energies.
8) Repeat from 2) for MC_{step} number of times, updating $\Gamma(t)$, $J^+(t)$ and $T(t)$ parameters for the SQPA copies

In this case, steps 2) to 8) constitute a single Monte Carlo step of the algorithm.

The i^{th} spin of the m^{th} Trotter of the s^{th} system copy flip, analogously to SQA, is accepted if ΔE is non-positive or if ΔE_{pot} is lower than 0 or with a probability equal to $e^{\frac{-\Delta E \cdot M}{T(s)}}$. The parameters evolution is monotonic with time as in SQA for the SQPA copies. The SQPA copies re-sampling is done by computing Q as a function of the energy of the best configuration of each copy, computing $N(E_i)$ for the i^{th} system copy and sampling the number of copies to be kept from a Poisson distribution centered in $N(E_i)$. For SQPT copies, the correlation factor J^+ considered in the ΔE_{kin} computation and the temperature T are not functions of the current iteration but of the considered system copy. Indeed, the algorithm SQPT parameters Γ, T and J^+ do not have a monotonic evolution with time, but their values change with time depending on the system copies energies according to the swap mechanism. A detailed and complete pseudo-code is reported in Algorithm 5.

Algorithm 5 Simulated Quantum Parallel Tempering - Population Annealing version 1

Input: **J** matrix and **h** vector
Output: Solution vector **sol** and **Energy** value

Initialize:
//Fill randomly spin configurations
//tensor $N \times M \times SYS$ with -1 or 1
//where $SYS = SYS_{\text{temp}} + SYS_{\text{pop}}$ is
//the total number of system copies
Randomly initialize **s_m** $\in \{-1, 1\}$
//Variables initialization
Energies $\leftarrow 0$
ΔE_local $\leftarrow 0$
//Vector of systems optimal energy
$E_{\text{sys\_opt}} \leftarrow 0$
//Precomputation of the involved Γ, T
//and J^+ parameters
$\Gamma \leftarrow \Gamma_0$
$T \leftarrow \frac{\text{MC\_step}}{1-\frac{1}{8}}$
$J^+ = \frac{T}{2} \log\left(\cot\left(\frac{\Gamma}{M \cdot T}\right)\right)$
//Precomputation of the involved Γ, T
//and J^+ parameters for SQPT
$\Gamma_{\text{list}} \leftarrow \text{linspace}\left(\Gamma_0, \Gamma_0\left(1 - \frac{\text{MC}_{\text{step}}}{\text{MC}_{\text{step}}}\right)\right), SYS_{\text{temp}}\right)$
$T_{\text{list}} \leftarrow \text{linspace}\left(\frac{\text{MC\_step}}{1-\frac{1}{8}}, \frac{\text{MC\_step}}{(1-\frac{1}{8})\cdot(\text{MC\_step})}, SYS_{\text{temp}}\right)$
$J^+_{\text{list}} \leftarrow \frac{T_{\text{list}}}{2} \log\left(\cot\left(\frac{\Gamma_{\text{list}}}{M \cdot T_{\text{list}}}\right)\right)$
//Systems time instant initialization
$s \leftarrow \text{linspace}(0, SYS_{\text{temp}} - 1, SYS_{\text{temp}})$
//Systems time instant initialization
//ΔE_local precomputation
//Current energy evaluation
for $sys = 0$ **To** $(SYS_{\text{temp}} + SYS_{\text{pop}}) - 1$ **do**
  for $m = 0$ **To** $M - 1$ **do**
    for $n = 0$ **To** $N - 1$ **do**
      for $k = 0$ **To** $N - 1$ **do**
        //Symmetric J matrix
        $\Delta\text{E\_local}_{n,m,sys} \mathrel{+}= 2 \cdot \text{J}_{n,k} \cdot \text{s\_m}_{m,n,sys}$
      end for
      $\Delta\text{E\_local}_{n,m,sys} \mathrel{+}= \text{h}_n$
    end for
    $\text{Energies}_{m,sys} = \text{s\_m}_{m,sys} \times (\text{J} \times \text{s\_m}^{\text{T}}_{m,sys})^{\text{T}}$
      $+\text{s\_m}_{m,sys} \times \text{h}^{\text{T}}$
    //Save current optimal solution
    if $m, n = 0$ **or** Energy $\geq \text{Energies}_{m,sys}$ **then**
      Energy $\leftarrow \text{Energies}_{m,sys}$
      sol $\leftarrow \text{s\_m}_m$
    end if
    //Save current optimal solution
    //system energy
    if $m = 0$ **or** $E_{\text{sys\_opt}_{sys}} \geq \text{Energies}_{m,sys}$ **then**
      $E_{\text{sys\_opt}_{sys}} \leftarrow \text{Energies}_{m,sys}$
    end if
  end for
end for

Monte Carlo steps:
for $t = 0$ **To** MC_step $- 1$ **do**
  for $sys = 0$ **To** $SYS_{\text{pop}} - 1$ **do**
    for $m = 0$ **To** $M - 1$ **do**
      //Identify Trotter neighbors
      Identify n_dx and n_sx //$m-1$ and $m+1$
      for $n = 0$ **To** $N - 1$ **do**
        //Evaluation of ΔE
        //for flipping the spin
        $\Delta\text{E\_pot} = -\Delta\text{E\_local}_{n,m,sys}$
          $\cdot(2 \cdot \text{s\_m}_{m,n,sys})$
        $\Delta\text{E\_kin} = J^+_{\text{list}}(s_{sys}) \cdot (\text{s\_m}_{\text{n\_dx},n,sys} +$
          $+\text{s\_m}_{\text{n\_sx},n,sys}) \cdot (-2 \cdot \text{s\_m}_{m,n,sys})$
        $\Delta\text{E} = \frac{\Delta\text{E\_pot}}{M} + \Delta\text{E\_kin}$
        $r = rand(0, 1)$
        //Metropolis-Hasting condition

if ΔE_pot < 0 **or** $r < e^{\frac{-\Delta E \cdot M}{T_{\text{list}}(s_{sys})}}$ **then**
//Spin-flip
s_m$_{m,n,sys} = -$s_m$_{m,n,sys}$
Energies$_{m,sys}$ = Energies$_{m,sys} + \Delta$E_pot
//Update optimal solution
if Energy $\geq$ Energies$_{m,sys}$ **then**
Energy $\leftarrow$ Energies$_{m,sys}$
sol $\leftarrow$ s_m$_{m,sys}$
end if
//Update ΔE_local
if $m = 0$ **or** $E_{\text{sys\_opt}_{sys}} \geq$ Energies$_{m,sys}$ **then**
$E_{\text{sys\_opt}_{sys}} \leftarrow$ Energies$_{m,sys}$
end if
for $k = 0$ **To** $N - 1$ **do**
//Symmetric J matrix
ΔE_local$_{k,m,sys}$ += $4 \cdot$ J$_{n,k}\cdot$ s_m$_{m,n,sys}$
end for
end if
end for
end for
end for
//Update the evolution parameters
for $sys_1 = 0$ **To** $SYS_{\text{pop}} - 1$ **do**
for $sys_2 = 0$ **To** $SYS_{\text{pop}} - 1$ **do**
Compute $T_{\text{eff}_{sys_1}}$
Compute $T_{\text{eff}_{sys_2}}$
//Metropolis-Hasting condition
if $r < e^{(\frac{1}{T_{\text{eff}_{sys_1}}} - \frac{1}{T_{\text{eff}_{sys_2}}})(E_{\text{sys\_opt}_{sys_1}} - E_{\text{sys\_opt}_{sys_2}})}$ **then**
//Time instants swap
$s_{sys_1} \leftrightarrow s_{sys_1}$
end if
end for
end for
for $sys = SYS_{\text{temp}}$ **To** $(SYS_{\text{pop}} + SYS_{\text{temp}}) - 1$ **do**
for $m = 0$ **To** $M - 1$ **do**
//Identify Trotter neighbors
Identify n_dx and n_sx //$m-1$ and $m+1$
for $n = 0$ **To** $N - 1$ **do**
//Evaluation of ΔE
//for flipping the spin
ΔE_pot $= -\Delta$E_local$_{n,m,sys}$ $\cdot(2 \cdot$ s_m$_{m,n,sys})$
ΔE_kin $= J^{+}_{\text{list}}(s_{sys}) \cdot (s_m_{\text{n_dx},n,sys} +$ $+$s_m$_{\text{n_sx},n,sys}) \cdot (-2 \cdot$ s_m$_{m,n,sys})$
$\Delta E = \frac{\Delta\text{E_pot}}{M} + \Delta$E_kin
$r = rand(0, 1)$
//Metropolis-Hasting condition
if ΔE_pot < 0 **or** $r < e^{\frac{-\Delta E \cdot M}{T_{\text{list}}(s_{sys})}}$ **then**
//Spin-flip
s_m$_{m,n,sys} = -$s_m$_{m,n,sys}$
Energies$_{m,sys}$ = Energies$_{m,sys} + \Delta$E_pot
//Update optimal solution
if Energy $\geq$ Energies$_{m,sys}$ **then**
Energy $\leftarrow$ Energies$_{m,sys}$
sol $\leftarrow$ s_m$_{m,sys}$
end if
//Update ΔE_local
if $m = 0$ **or** $E_{\text{sys\_opt}_{sys}} \geq$ Energies$_{m,sys}$ **then**
$E_{\text{sys\_opt}_{sys}} \leftarrow$ Energies$_{m,sys}$
end if
for $k = 0$ **To** $N - 1$ **do**
//Symmetric J matrix
ΔE_local$_{k,m,sys}$ += $4 \cdot$ J$_{n,k}\cdot$ s_m$_{m,n,sys}$
end for
end if
end for
end for
end for
//re-sampling
$Q \leftarrow 0$
$\Gamma_{\text{next}} = \Gamma_0 \cdot (1 - \frac{t+1}{\text{MC\_step}+1})$
for $sys = SYS_{\text{temp}}$ **To** $(SYS_{\text{pop}} + SYS_{\text{temp}}) - 1$ **do**
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma^2+1}}{\Gamma} + \frac{1}{\Gamma}\right|^2\right)\right]^{-1}$
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma^2_{\text{next}}+1}}{\Gamma_{\text{next}}} + \frac{1}{\Gamma_{\text{next}}}\right|^2\right)\right]^{-1}$
$Q+ = \exp\left(\left(\frac{1}{T_{\text{eff1}}} - \frac{1}{T_{\text{eff2}}}\right)E_{\text{sys\_opt}_{sys}}\right)$
end for
$Q = \frac{Q}{SYS_{\text{pop}}}$
//declare temporary tensor $N \times M \times SYS$
s_m$_{\text{temp}}$
$E_{\text{sys\_opt}_{\text{temp}}}$
ΔE_local$_{\text{temp}}$
for $sys = 0$ **To** $SYS_{\text{temp}} - 1$ **do**
s_m$_{\text{temp}_{sys}}$ = s_m$_{sys}$
$E_{\text{sys\_opt}_{\text{temp}_{sys}}} = E_{\text{sys\_opt}_{sys}}$
ΔE_local$_{\text{temp}_{sys}} = \Delta$E_local$_{sys}$
end for
//Variable for counting copies
$c \leftarrow SYS_{\text{temp}}$
for $sys = SYS_{\text{temp}}$ **To** $(SYS_{\text{pop}} + SYS_{\text{temp}}) - 1$ **do**
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma^2+1}}{\Gamma} + \frac{1}{\Gamma}\right|^2\right)\right]^{-1}$
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma^2_{\text{next}}+1}}{\Gamma_{\text{next}}} + \frac{1}{\Gamma_{\text{next}}}\right|^2\right)\right]^{-1}$
$N_{\text{mean}} = \frac{1}{Q}\exp\left(\left(\frac{1}{T_{\text{eff1}}} - \frac{1}{T_{\text{eff1}}}\right)E_{\text{sys\_opt}_{sys}}\right)$
$R = \text{Poisson}(N)$
for $r = 0$ **To** $R - 1$ **do**
//For maintaining constant the
//number of system
if $c < (SYS_{\text{pop}} + SYS_{\text{temp}})$ **then**
s_m$_{\text{temp}_c}$ = s_m$_{sys}$
$E_{\text{sys\_opt}_{\text{temp}_c}} = E_{\text{sys\_opt}_{sys}}$
ΔE_local$_{\text{temp}_c} = \Delta$E_local$_{sys}$
$c+ = 1$
end if
end for
end for
//For maintaining constant the number
//of system
while $c < (SYS_{\text{pop}} + SYS_{\text{temp}})$ **do**
s_m$_{\text{temp}_c}$ = s_m$_{SYS-1}$
$E_{\text{sys\_opt}_{\text{temp}_c}} = E_{\text{sys\_opt}_{SYS-1}}$
ΔE_local$_{\text{temp}_c} = \Delta$E_local$_{\text{temp}}$
$c+ = 1$
end while
s_m = s_m$_{\text{temp}}$
$E_{\text{sys\_opt}} = E_{\text{sys\_opt}_{\text{temp}}}$
ΔE_local = ΔE_local$_{\text{temp}}$
//Update the evolution parameters
$\Gamma = \Gamma_0 \cdot (1 - \frac{t}{\text{MC\_step}+1})$
$T = \frac{\text{MC\_step}}{(1-\frac{1}{8})\cdot(t+1)}$
$J^{+} = \frac{T}{2}\log\left(\cot\left(\frac{\Gamma}{M \cdot T}\right)\right)$

end for

Return: Energy, sol

A further advantage of the presented approach is that the spins update of each system copy can be done completely in parallel to the other without delays. In this way, the time required for the algorithm execution is:

$$t_{\text{exe\_parallel}} = t_{\text{init}} + ((M-1)\cdot t_{\text{spin}} + N \cdot t_{\text{spin}} + \\ + \max\left(t_{\text{re-sampling}}, t_{\text{swap}}\right)) \cdot \text{MC\_step}\,, \quad (66)$$

where t_{init} is the initialization time, t_{spin} is the time required for the update of a single spin (inner loop), $t_{\text{re-sampling}}$ is the one required for re-sampling system copies and t_{swap} is the one required for swapping the system copies time instant, instead of:

$$t_{\text{exe\_serial}} = t_{\text{init}} + ((M \cdot N \cdot (SYS_{\text{pop}} + SYS_{\text{temp}})) \cdot t_{\text{spin}} + \\ + t_{\text{re-sampling}} + t_{\text{swap}}) \cdot \text{MC\_step}\,. \quad (67)$$

The savings is equal to $(M \cdot N \cdot (SYS_{\text{pop}} + SYS_{\text{temp}}) - M + 1 - N) \cdot t_{\text{spin}} \cdot \text{MC\_step}$. By comparing $t_{\text{exe\_parallel}}$ with one of SQA in the same condition, i.e. with the same total number of Ising copies ($M_{\text{SQA}} = M_{\text{SQPTPA1}} \cdot (SYS_{\text{pop}_{\text{SQPTPA1}}} + SYS_{\text{temp}_{\text{SQPTPA1}}})$), for spins update, the saving is equal to $(M_{\text{SQPT}} \cdot ((SYS_{\text{pop}_{\text{SQPTPA1}}} + SYS_{\text{temp}_{\text{SQPTPA1}}}) - 1) \cdot t_{\text{spin}}) \cdot \text{MC\_step}$. From the spins update time point of view, considering the same operating condition, i.e. with the same total number of Ising copies ($M_{\text{SQPTPA1}} \cdot (SYS_{\text{pop}_{\text{SQPTPA1}}} + SYS_{\text{temp}_{\text{SQPTPA1}}}) = M_{\text{SQPT}} \cdot SYS_{\text{SQPT}} = M_{\text{SQPA}} \cdot SYS_{\text{SQPA}}$), the SQPTPA1, the SQPT and the SQPA are perfectly equivalent, indeed, in all cases, it is possible to perfectly parallelize the system copies evaluations. In a hardware implementation, this time could compensate for the overhead related to the re-sampling and swapping time, which will be higher than the parameter update time of SQA, especially if fast solutions will be found. Therefore, in this work, the iterations of the algorithms are considered comparable.

D. SIMULATED QUANTUM PARALLEL TEMPERING - POPULATION ANNEALING VERSION 2

Simulated Quantum Parallel Tempering - Population Annealing version 2 (**SQPTPA2**) was proposed to join the advantages of SQPT and SQPA algorithms to obtain a solver as flexible as possible. It was obtained substantially by running at the same time both the SQPT and the SQPA, but with a one system copy shared between the two approaches. This means that the spins configurations of the last SQPT system copy are considered in the evaluation of the number of copies to be kept of each system copy-spins configuration of SQPA.
In particular, a system of SYS_{temp} plus SYS_{pop} copies composed of SQA Trotter systems was considered. The SYS_{temp} copies evolve according to the SQPT algorithm, while the SYS_{pop} copies plus the last SQPT follow the SQPA one. This is different from running the two algorithms in parallel and choosing the optimal solution with a majority voting mechanism in that the two algorithms can influence each other through the shared system copy. In this way, if SQPT is exploring a more promising region of the energy profile, it can guide the search of SQPA towards that area. *Vice versa*, if SQPA obtains lower energy spins configurations than SQPT, it can orient SQPT exploration. The main algorithm evolution steps are the following:

1) For each Trotter of each system, an arbitrary solution is considered.
2) At a given time instant t, a transverse field $\Gamma(t)$, a correlation factor $J^{+}(t)$ and a temperature parameter $T(t)$ are assigned to each SQPT system copy from a uniformly distributed list in range 0 to $\text{MC}_{\text{step}} - 1$.
3) For each spin of each replica of each system copy, an update is considered and accepted according to the MH algorithm.
4) Evaluate which replica gives the current best solution and if this solution is better than the ones obtained in the previous step.
5) Save the best current solution of each system copy.
6) Evaluate for each couple of SQPT system copies the time instants swap according to the MH algorithm with an acceptance rate equal to Equation 58.
7) **re-sampling** the SQPA system copies plus the last SQPT according to their energies.
8) Repeat from 2) for MC_{step} number of times, updating $\Gamma(t)$, $J^{+}(t)$ and $T(t)$ parameters for the SQPA copies

In this case, steps 2) to 8) constitute a single Monte Carlo step of the algorithm.
The flip of the i^{th} spin of the m^{th} Trotter of the s^{th} system copy is accepted, analogously to SQA, if ΔE is non-positive or if ΔE_{pot} is lower than 0 or with a probability equal to $e^{\frac{-\Delta E \cdot M}{T(s)}}$. The parameters evolution is monotonic with time as in SQA for the SQPA copies. The SQPA copies re-sampling is done by computing Q as a function of the energy of the best configuration of each copy, computing $N(E_i)$ for the i^{th} system copy and sampling the number of copies to be kept from a Poisson distribution centered in $N(E_i)$. For SQPT copies, the correlation factor J^{+} considered in the ΔE_{kin} computation and the temperature T are not functions of the current iteration but of the considered system copy, in that these parameters do not have a monotonic evolution with time, but ones depending on the parameters swap mechanism. A detailed and complete pseudo-code is reported in Algorithm 6.

Algorithm 6 Simulated Quantum Parallel Tempering - Population Annealing version 2

Input: **J** matrix and **h** vector
Output: Solution vector **sol** and **Energy** value

Initialize:
//Fill randomly spin configurations
//tensor $N \times M \times SYS$ with -1 or 1
//where $SYS = SYS_{\text{temp}} + SYS_{\text{pop}}$ is
//the total number of system copies
Randomly initialize **s_m** $\in \{-1, 1\}$

```
//Variables initialization
Energies ← 0
ΔE_local ← 0
//Vector of systems optimal energy
E_sys_opt ← 0
//Precomputation of the involved Γ, T
//and J+ parameters
Γ ← Γ_0
T ← MC_step / (1 − 1/8)
J+ = (T/2) log(cot(Γ/(M·T)))
//Precomputation of the involved Γ, T
//and J+ parameters for SQPT
Γ_list ← linspace(Γ_0, Γ_0(1 − MC_step/MC_step)), SYS_temp)
T_list ← linspace(MC_step/(1 − 1/8), MC_step/((1 − 1/8)·(MC_step)), SYS_temp)
J+_list ← (T_list/2) log(cot(Γ_list/(M·T_list)))
//Systems time instant initialization
s ← linspace(0, SYS_temp − 1, SYS_temp)
//Systems time instant initialization
//ΔE_local precomputation
//Current energy evaluation
for sys = 0 To (SYS_temp + SYS_pop) − 1 do
    for m = 0 To M − 1 do
        for n = 0 To N − 1 do
            for k = 0 To N − 1 do
                //Symmetric J matrix
                ΔE_local_{n,m,sys} += 2 · J_{n,k} · s_m_{m,n,sys}
            end for
            ΔE_local_{n,m,sys} += h_n
        end for
        Energies_{m,sys} = s_m_{m,sys} × (J × s_m^T_{m,sys})^T
                +s_m_{m,sys} × h^T
        //Save current optimal solution
        if m, n = 0 or Energy ≥ Energies_{m,sys} then
            Energy ← Energies_{m,sys}
            sol ← s_m_m
        end if
        //Save current optimal solution
        //system energy
        if m = 0 or E_sys_opt_sys ≥ Energies_{m,sys} then
            E_sys_opt_sys ← Energies_{m,sys}
        end if
    end for
end for

Monte Carlo steps:
for t = 0 To MC_step − 1 do
    for sys = 0 To SYS_pop − 1 do
        for m = 0 To M − 1 do
            //Identify Trotter neighbors
            Identify n_dx and n_sx //m − 1 and m + 1
            for n = 0 To N − 1 do
                //Evaluation of ΔE
                //for flipping the spin
                ΔE_pot = −ΔE_local_{n,m,sys}
                        ·(2 · s_m_{m,n,sys})
                ΔE_kin = J+_list(s_sys) · (s_m_{n_dx,n,sys}+
                        +s_m_{n_sx,n,sys}) · (−2 · s_m_{m,n,sys})
                ΔE = ΔE_pot/M + ΔE_kin
                r = rand(0, 1)
                //Metropolis-Hasting condition
                if ΔE_pot < 0 or r < e^{−ΔE·M / T_list(s_sys)} then
                    //Spin-flip
                    s_m_{m,n,sys} = −s_m_{m,n,sys}
                    Energies_{m,sys} = Energies_{m,sys} + ΔE_pot
                    //Update optimal solution
                    if Energy ≥ Energies_{m,sys} then
                        Energy ← Energies_{m,sys}
                        sol ← s_m_{m,sys}
                    end if
                    //Update ΔE_local
                    if m = 0 or E_sys_opt_sys ≥ Energies_{m,sys} then
                        E_sys_opt_sys ← Energies_{m,sys}
                    end if
                    for k = 0 To N − 1 do
                        //Symmetric J matrix
                        ΔE_local_{k,m,sys} += 4 · J_{n,k} ·
                                s_m_{m,n,sys}
                    end for
                end if
            end for
        end for
    end for
end for
//Update the evolution parameters
for sys_1 = 0 To SYS_pop − 1 do
    for sys_2 = 0 To SYS_pop − 1 do
        Compute T_eff_{sys_1}
        Compute T_eff_{sys_2}
        //Metropolis-Hasting condition
        if r < e^{(1/T_eff_{sys_1} − 1/T_eff_{sys_2})(E_sys_opt_{sys_1} − E_sys_opt_{sys_2})} then
            //Time instants swap
            s_{sys_1} ↔ s_{sys_1}
        end if
    end for
end for
for sys = SYS_temp To (SYS_pop + SYS_temp) − 1 do
    for m = 0 To M − 1 do
        //Identify Trotter neighbors
        Identify n_dx and n_sx //m − 1 and m + 1
        for n = 0 To N − 1 do
            //Evaluation of ΔE
            //for flipping the spin
            ΔE_pot = −ΔE_local_{n,m,sys}
                    ·(2 · s_m_{m,n,sys})
            ΔE_kin = J+_list(s_sys) · (s_m_{n_dx,n,sys}+
                    +s_m_{n_sx,n,sys}) · (−2 · s_m_{m,n,sys})
            ΔE = ΔE_pot/M + ΔE_kin
            r = rand(0, 1)
            //Metropolis-Hasting condition
            if ΔE_pot < 0 or r < e^{−ΔE·M / T_list(s_sys)} then
                //Spin-flip
                s_m_{m,n,sys} = −s_m_{m,n,sys}
                Energies_{m,sys} = Energies_{m,sys} + ΔE_pot
                //Update optimal solution
                if Energy ≥ Energies_{m,sys} then
                    Energy ← Energies_{m,sys}
                    sol ← s_m_{m,sys}
                end if
                //Update ΔE_local
                if m = 0 or E_sys_opt_sys ≥ Energies_{m,sys} then
                    E_sys_opt_sys ← Energies_{m,sys}
                end if
                for k = 0 To N − 1 do
                    //Symmetric J matrix
                    ΔE_local_{k,m,sys} += 4 · J_{n,k} ·
                            s_m_{m,n,sys}
                end for
```

end if
end for
end for
end for
//re-sampling
$Q \leftarrow 0$ re-sampling $sys = SYS_{\text{temp}} - 1$
$\Gamma_{\text{next}} = \Gamma_{\text{list}}(s_{sys}) - \Gamma_0 \cdot (\frac{1}{\text{MC\_step}+1})$
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma_{\text{list}}(s_{sys})^2+1}}{\Gamma_{\text{list}}(s_{sys})} + \frac{1}{\Gamma_{\text{list}}(s_{sys})}\right|^2\right)\right]^{-1}$
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma_{\text{next}}^2+1}}{\Gamma_{\text{next}}} + \frac{1}{\Gamma_{\text{next}}}\right|^2\right)\right]^{-1}$
$Q+ = \exp\left(\left(\frac{1}{T_{\text{eff1}}} - \frac{1}{T_{\text{eff2}}}\right)E_{\text{sys\_opt}_{sys}}\right)$
$\Gamma_{\text{next}} = \Gamma_0 \cdot (1 - \frac{t+1}{\text{MC\_step}+1})$
for $sys = SYS_{\text{temp}}$ **To** $(SYS_{\text{pop}} + SYS_{\text{temp}}) - 1$ **do**
  $T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma^2+1}}{\Gamma} + \frac{1}{\Gamma}\right|^2\right)\right]^{-1}$
  $T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma_{\text{next}}^2+1}}{\Gamma_{\text{next}}} + \frac{1}{\Gamma_{\text{next}}}\right|^2\right)\right]^{-1}$
  $Q+ = \exp\left(\left(\frac{1}{T_{\text{eff1}}} - \frac{1}{T_{\text{eff2}}}\right)E_{\text{sys\_opt}_{sys}}\right)$
end for
$Q = \frac{Q}{SYS_{\text{pop}}+1}$
//declare temporary tensor $N \times M \times SYS$
s_m_{temp}
$E_{\text{sys\_opt}_{\text{temp}}}$
$\Delta\text{E\_local}_{\text{temp}}$
for $sys = 0$ **To** $SYS_{\text{temp}} - 2$ **do**
  $\text{s\_m}_{\text{temp}_{sys}} = \text{s\_m}_{sys}$
  $E_{\text{sys\_opt}_{\text{temp}_{sys}}} = E_{\text{sys\_opt}_{sys}}$
  $\Delta\text{E\_local}_{\text{temp}_{sys}} = \Delta\text{E\_local}_{sys}$
end for
//Variable for counting copies
$c \leftarrow SYS_{\text{temp}} - 1$ re-sampling $sys = SYS_{\text{temp}} - 1$
$\Gamma_{\text{next}} = \Gamma_{\text{list}}(s_{sys}) - \Gamma_0 \cdot (\frac{1}{\text{MC\_step}+1})$
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma_{\text{list}}(s_{sys})^2+1}}{\Gamma_{\text{list}}(s_{sys})} + \frac{1}{\Gamma_{\text{list}}(s_{sys})}\right|^2\right)\right]^{-1}$
$T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma_{\text{next}}^2+1}}{\Gamma_{\text{next}}} + \frac{1}{\Gamma_{\text{next}}}\right|^2\right)\right]^{-1}$
$N_{\text{mean}} = \frac{1}{Q}\exp\left(\left(\frac{1}{T_{\text{eff1}}} - \frac{1}{T_{\text{eff1}}}\right)E_{\text{sys\_opt}_{sys}}\right)$
$R = \text{Poisson}(N)$
for $r = 0$ **To** $R - 1$ **do**
  //For maintaining constant the
  //number of system
  if $c < (SYS_{\text{pop}} + SYS_{\text{temp}})$ **then**
    $\text{s\_m}_{\text{temp}_c} = \text{s\_m}_{sys}$
    $E_{\text{sys\_opt}_{\text{temp}_c}} = E_{\text{sys\_opt}_{sys}}$
    $\Delta\text{E\_local}_{\text{temp}_c} = \Delta\text{E\_local}_{sys}$
    $c+ = 1$
  end if
end for
$\Gamma_{\text{next}} = \Gamma_0 \cdot (1 - \frac{t+1}{\text{MC\_step}+1})$
for $sys = SYS_{\text{temp}}$ **To** $(SYS_{\text{pop}} + SYS_{\text{temp}}) - 1$ **do**
  $T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma^2+1}}{\Gamma} + \frac{1}{\Gamma}\right|^2\right)\right]^{-1}$
  $T_{\text{eff1}} = 2\left[\ln\left(\left|\frac{\sqrt{\Gamma_{\text{next}}^2+1}}{\Gamma_{\text{next}}} + \frac{1}{\Gamma_{\text{next}}}\right|^2\right)\right]^{-1}$
  $N_{\text{mean}} = \frac{1}{Q}\exp\left(\left(\frac{1}{T_{\text{eff1}}} - \frac{1}{T_{\text{eff1}}}\right)E_{\text{sys\_opt}_{sys}}\right)$
  $R = \text{Poisson}(N)$
  for $r = 0$ **To** $R - 1$ **do**
    //For maintaining constant the
    //number of system
    if $c < (SYS_{\text{pop}} + SYS_{\text{temp}})$ **then**
      $\text{s\_m}_{\text{temp}_c} = \text{s\_m}_{sys}$
      $E_{\text{sys\_opt}_{\text{temp}_c}} = E_{\text{sys\_opt}_{sys}}$
      $\Delta\text{E\_local}_{\text{temp}_c} = \Delta\text{E\_local}_{sys}$
      $c+ = 1$
    end if
  end for
end for
//For maintaining constant the number
//of system
while $c < (SYS_{\text{pop}} + SYS_{\text{temp}})$ **do**
  $\text{s\_m}_{\text{temp}_c} = \text{s\_m}_{SYS-1}$
  $E_{\text{sys\_opt}_{\text{temp}_c}} = E_{\text{sys\_opt}_{SYS-1}}$
  $\Delta\text{E\_local}_{\text{temp}_c} = \Delta\text{E\_local}_{\text{temp}}$
  $c+ = 1$
end while
$\text{s\_m} = \text{s\_m}_{\text{temp}}$
$E_{\text{sys\_opt}} = E_{\text{sys\_opt}_{\text{temp}}}$
$\Delta\text{E\_local} = \Delta\text{E\_local}_{\text{temp}}$
//Update the evolution parameters
$\Gamma = \Gamma_0 \cdot (1 - \frac{t}{\text{MC\_step}+1})$
$T = \frac{\text{MC\_step}}{(1-\frac{1}{8})\cdot(t+1)}$
$J^+ = \frac{T}{2}\log\left(\cot\left(\frac{\Gamma}{M \cdot T}\right)\right)$
end for

Return: Energy, sol

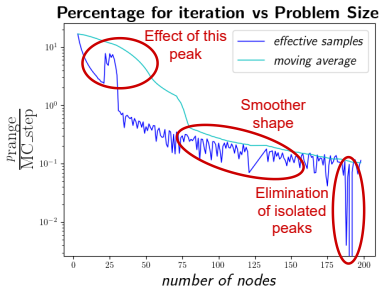
A further advantage of the presented approach is that the spins update of each system copy can be done completely in parallel to the other without delays. In this way, the time required for the algorithm execution is:

$$t_{\text{exe\_parallel}} = t_{\text{init}} + ((M-1)\cdot t_{\text{spin}} + N \cdot t_{\text{spin}} + \\ + \max(t_{\text{re-sampling}}, t_{\text{swap}})) \cdot \text{MC\_step}\,, \quad (68)$$

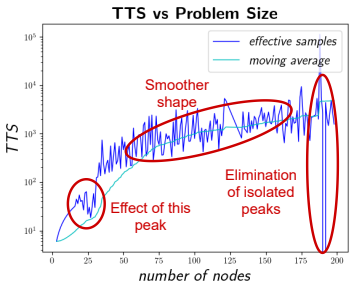
where t_{init} is the initialization time, t_{spin} is the time required for the update of a single spin (inner loop), $t_{\text{re-sampling}}$ is the one required for re-sampling system copies and t_{swap} is the one required for swapping the system copies time instant. On the other hand, the execution time for sequential execution is:

$$t_{\text{exe\_serial}} = t_{\text{init}} + ((M \cdot N \cdot (SYS_{\text{pop}} + SYS_{\text{temp}})) \cdot t_{\text{spin}} + \\ + t_{\text{re-sampling}} + t_{\text{swap}}) \cdot \text{MC\_step}\,. \quad (69)$$

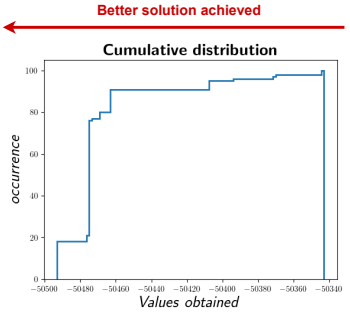
The savings is equal to $(M \cdot N \cdot (SYS_{\text{pop}} + SYS_{\text{temp}}) - M + 1 - N) \cdot t_{\text{spin}} \cdot \text{MC\_step}$. By comparing $t_{\text{exe\_parallel}}$ with one of SQA in the same condition, i.e. with the same total number of Ising copies ($M_{\text{SQA}} = M_{\text{SQPTPA2}} \cdot (SYS_{\text{pop}_{\text{SQPTPA2}}} + SYS_{\text{temp}_{\text{SQPTPA2}}})$), for spins update, the saving is equal to $(M_{\text{SQPT}} \cdot ((SYS_{\text{pop}_{\text{SQPTPA2}}} + SYS_{\text{temp}_{\text{SQPTPA2}}}) - 1) \cdot t_{\text{spin}}) \cdot \text{MC\_step}$. From the spins update time point of view, considering the same operating condition, i.e. with the same total number of Ising copies ($M_{\text{SQPTPA2}} \cdot (SYS_{\text{pop}_{\text{SQPTPA2}}} + SYS_{\text{temp}_{\text{SQPTPA2}}}) = M_{\text{SQPT}} \cdot SYS_{\text{SQPT}} = M_{\text{SQPA}} \cdot SYS_{\text{SQPA}}$), the SQPTPA2, the SQPT and the SQPA are perfectly equivalent, indeed, in all cases, it is possible to perfectly parallelize the system copies evaluations. In a hardware implementation, this time could compensate for the overhead related to the re-sampling and swapping time, which will be higher than the



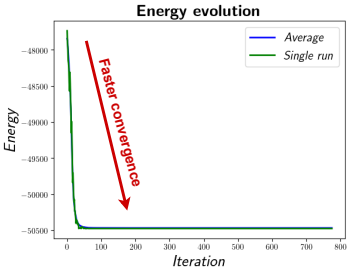
(a) Effect of moving average with a w equal to 50 in the $\frac{p_{\text{range}}}{\text{MC\_step}}$ evolution varying the number of the involved binary variables



(b) Effect of moving average with a w equal to 50 in the time-to-solution (TTS) evolution varying the number of the involved binary variables



(c) An example of a cumulative distribution obtained running one hundred times a solver with the same problem



(d) An example of the energy evolution of a single run and the average energy evolution of one hundred runs of a solver

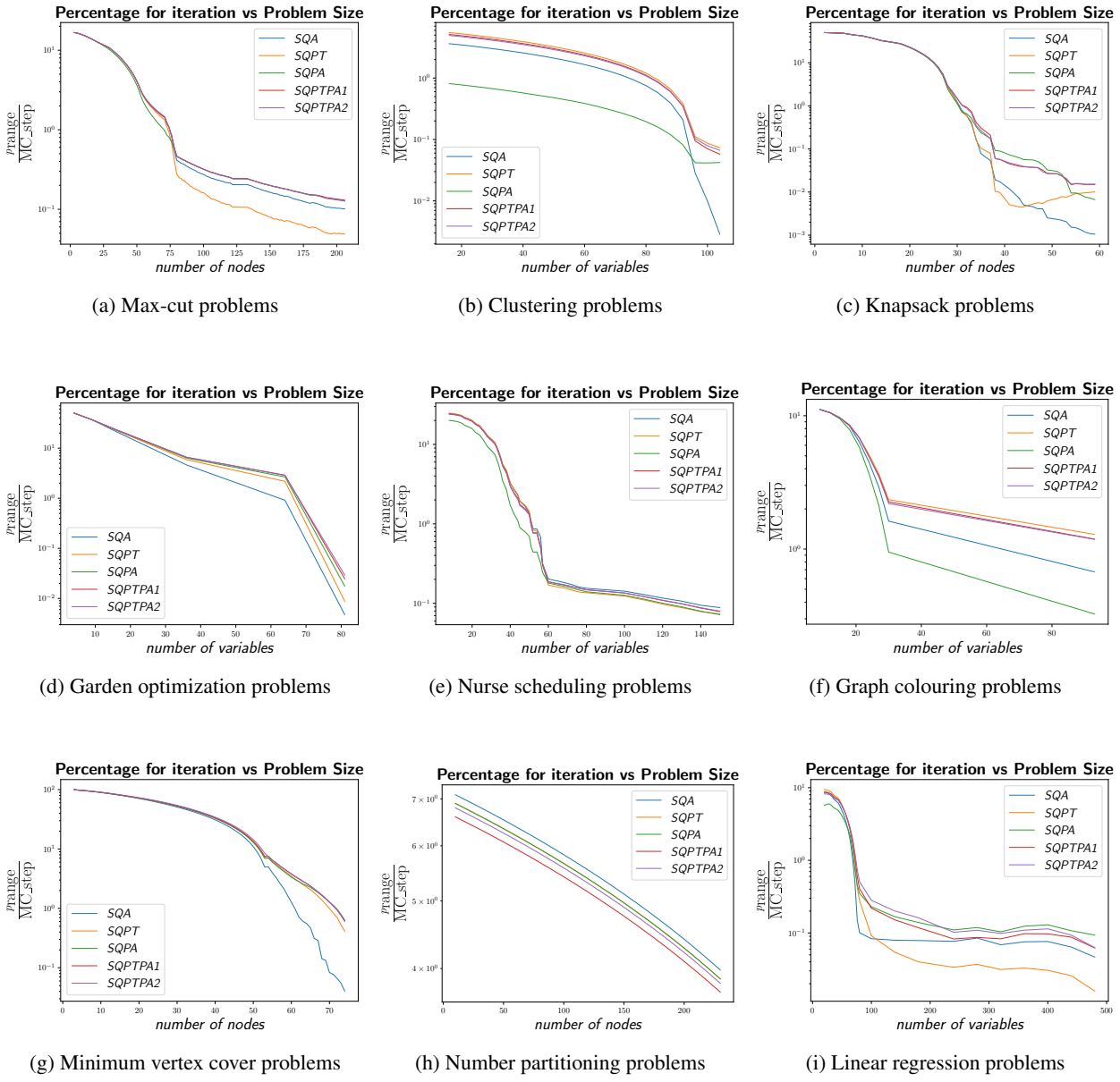
FIGURE 13: Explanation through examples of the graphical representation of figures of merit. In Figure 13a shows the effect of moving average on $\frac{p_{\text{range}}}{\text{MC\_step}}$ evolution varying the number of the involved binary variables. It is possible to notice that it smooths the profile and removes isolated peaks, thus allowing easier recognition of the trend. It has the same effect on the time-to-solution (TTS) evolution varying the number of the involved binary variables, as reported in Figure 13b. Figure 13c shows a cumulative distribution, obtained by running one hundred times a solver on the same problem. In order to understand the meaning of this, one rule has to be considered: 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 objective function are located. Finally, Figure 13d reports the energy evolution of a single run and the average energy evolution of one hundred runs of a solver. In this case, the higher the derivative of the evolution on the left, the faster the convergence to the final energy, which can be the optimal one.

parameter update time of SQA, especially if fast solutions will be found. Therefore, in this work, the iterations of the algorithms are considered comparable.

The only time difference between the SQPTPA1 and SQPTPA2 algorithms from the execution time point of view is that the $t_{\text{re-sampling}}$ of the second one is longer than the ones of the first for the same number of SQPA and SQPT systems. This is because SQPTPA2 has to compute the number of copies to be kept also for the last SQPT copy. However, for a sufficiently high number of systems, the time difference is negligible. For the sake of simplicity, the total execution time of the two algorithms was considered comparable.

IV. RESULTS

This section reports the results associated with the most complex benchmark problems (Section II-A4) solved with the SQA, the SQPT, the SQPA, the SQPTPA1 and the SQPTPA2 algorithms. All the other results not reported in the following are available in the supplementary information file, with the same format, in terms of plots and tables. The codes



(a) Max-cut problems (b) Clustering problems (c) Knapsack problems

(d) Garden optimization problems (e) Nurse scheduling problems (f) Graph colouring problems

(g) Minimum vertex cover problems (h) Number partitioning problems (i) Linear regression problems

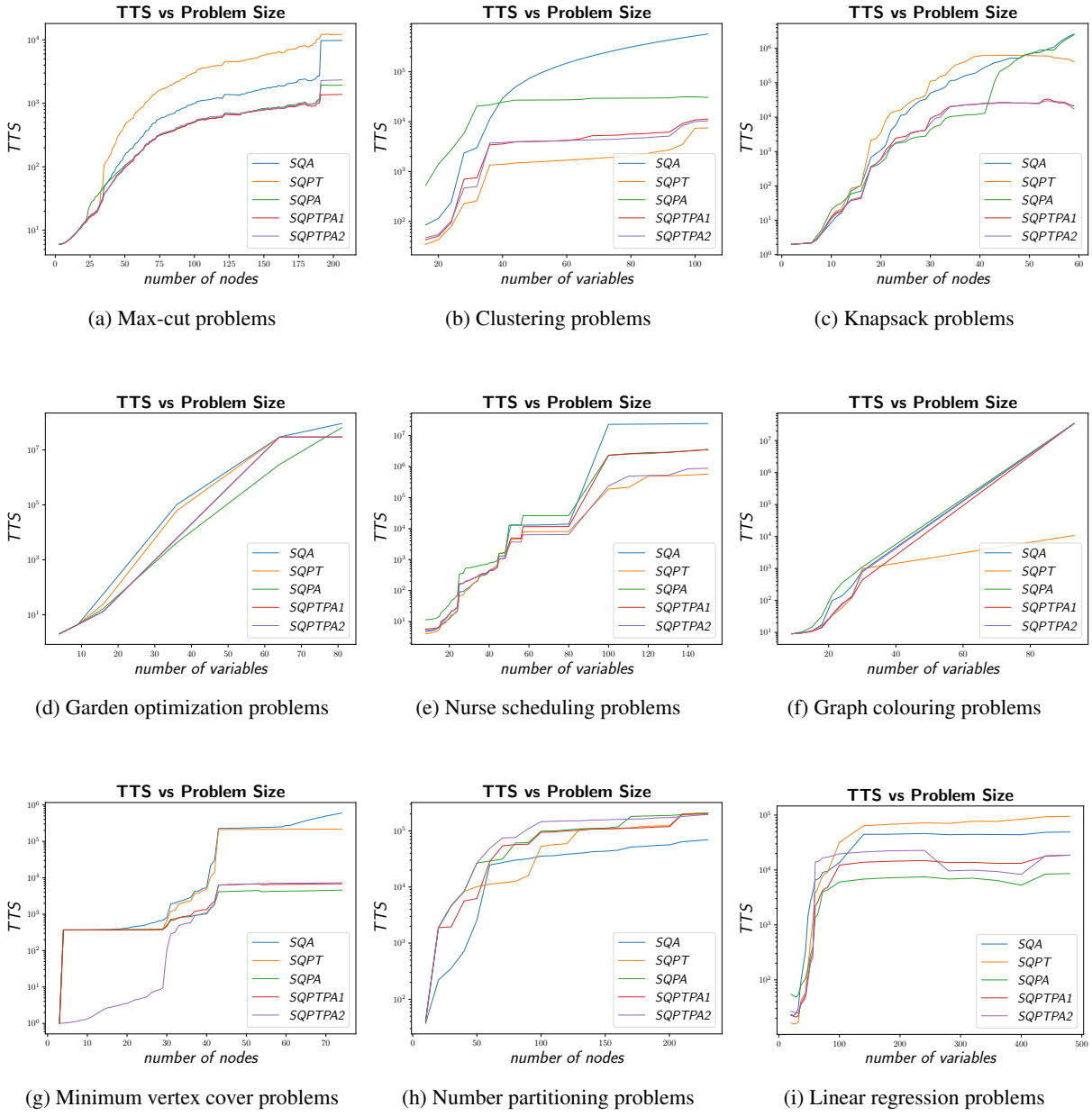
FIGURE 14: Evolution of $\frac{p_{\text{range}}}{\text{MC\_step}}$ varying the number of involved binary variables in the optimization problems.

employed for obtaining the reported results are available in the GitHub repository.

A. SETUP OVERVIEW

All the tests were performed by exploiting **Python** implementations of SQA, an SQPT, an SQPA, an SQPTPA1 and SQPTPA2. The same programming language was chosen in all cases to test the algorithms in equivalent conditions. Moreover, Python provides many libraries for describing optimization problems in QUBO formalism, such as qubovert, which was exploited for generating benchmark problems in this work.

Each solver was implemented as a Python class with proper methods for setting the degrees of freedom of the algorithm, such as the number of iterations and the number of runs, for executing the algorithm and for writing report files. The developed solver classes must be seen as **proof of concept** or **software models** for hardware accelerators, which could be developed in the future. Indeed, the potential of the proposed algorithms is expected to be entirely appreciated only with a



(a) Max-cut problems (b) Clustering problems (c) Knapsack problems

(d) Garden optimization problems (e) Nurse scheduling problems (f) Graph colouring problems

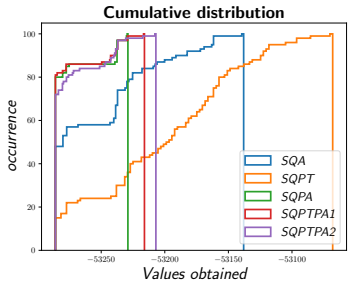
(g) Minimum vertex cover problems (h) Number partitioning problems (i) Linear regression problems

FIGURE 15: Evolution of time-to-solution (TTS) varying the number of involved binary variables in the optimization problems.

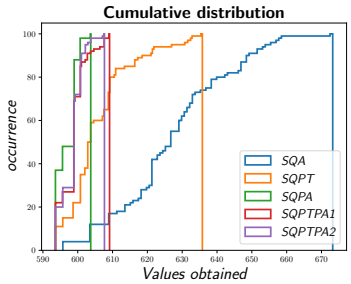
hardware implementation, capable of parallelizing the computation as much as possible.

Tests were executed on a single-process Intel(R) Xeon(R) Gold 6134 CPU @ 3.20 GHz opta-core, Model 85, with a memory of about 103 GB [95]. Each analyzed solver was executed on the same optimization problem one hundred times to extract statistics on its effectiveness in solving it. To better compare the results, the same number of iterations, the same initial transverse-field-temperature pair ($\Gamma_0 = 1$, $T_0 = 1$), and the same total number of copies ($M_{SQA} = M_{\text{SQPT}} \cdot SYS_{\text{SQPT}} = M_{\text{SQPA}} \cdot SYS_{\text{SQPA}} = M_{\text{SQPTPA1}} \cdot (SYS_{\text{pop}_{\text{SQPTPA1}}} + SYS_{\text{temp}_{\text{SQPTPA1}}}) = M_{\text{SQPTPA2}} \cdot (SYS_{\text{pop}_{\text{SQPTPA2}}} + SYS_{\text{temp}_{\text{SQPTPA2}}})$) were considered for each solver in each test.

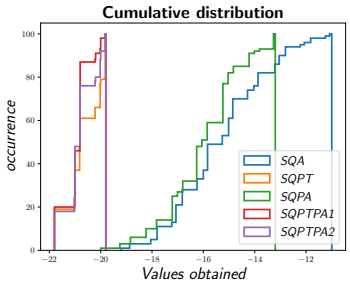
The benchmark problems were automatically generated by starting from a generic QUBO description, defining the size and randomly generating some elements of the problems, such as the weights of edges in the max-cut problem, the



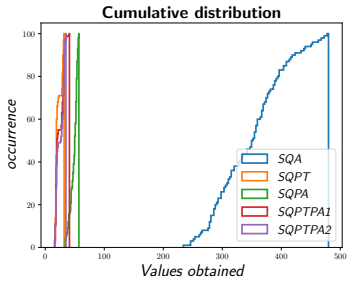
(a) Cumulative distribution of a 200-node max-cut problem obtained considering 18 total copies and 800 MC_step



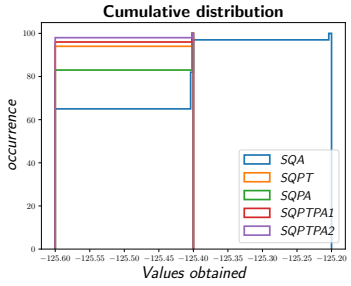
(b) Cumulative distribution of a 25-data 4-cluster (100-variable) clustering problem obtained considering 32 total copies and 2000 MC_step



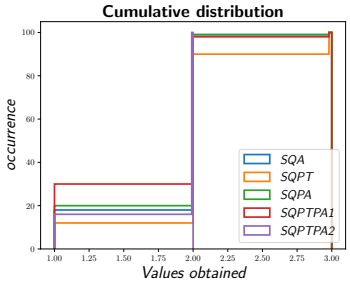
(c) Cumulative distribution of a 51-variable clustering problem obtained considering 32 total copies and 5100 MC_step



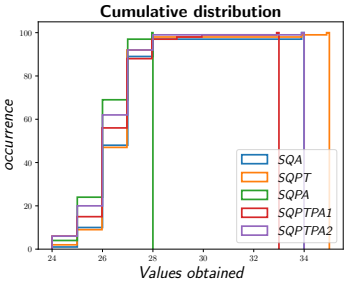
(d) Cumulative distribution of a 9-pot and 9-plant (81-variable) garden optimization problem obtained considering 32 total copies and 40500 MC_step



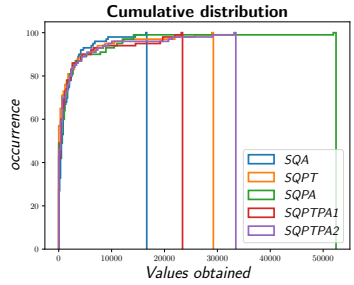
(e) Cumulative distribution of a 4-nurse and 20-day (80-variable) nurse scheduling problem obtained considering 18 total copies and 800 MC_step



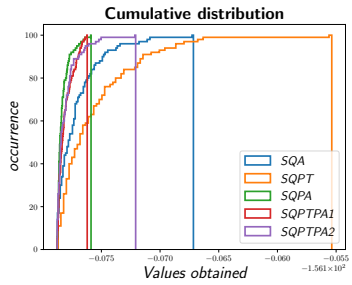
(f) Cumulative distribution of a 10-node and 3-colour (30-variable) graph colouring problem obtained considering 18 total copies and 120 MC_step



(g) Cumulative distribution of a 31-node minimum vertex cover problem obtained considering 18 total copies and 124 MC_step

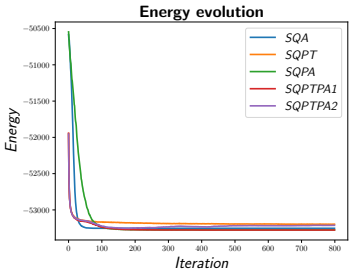


(h) Cumulative distribution of a 200-node number partitioning problem obtained considering 18 total copies and 800 MC_step

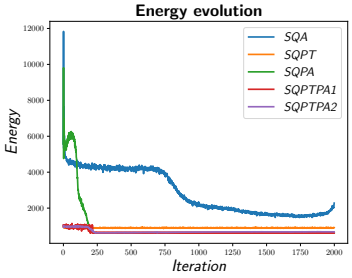


(i) Cumulative distribution of a 100-power (400-variable) linear regression problem obtained considering 18 total copies and 200 MC_step

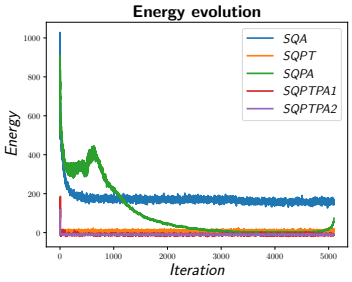
FIGURE 16: Cumulative distributions obtained by running one hundred times each solver for each optimization



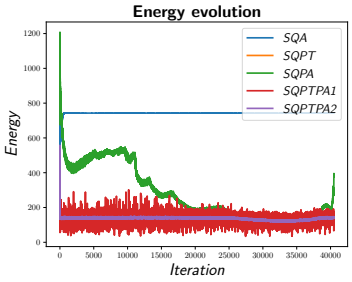
(a) Energy evolution of a 200-node max-cut problem obtained considering 18 total copies and 800 MC_step



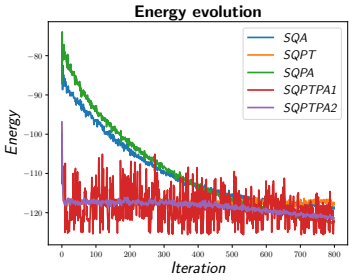
(b) Energy evolution of a 25-data 4-cluster (100-variable) clustering problem obtained considering 32 total copies and 2000 MC_step



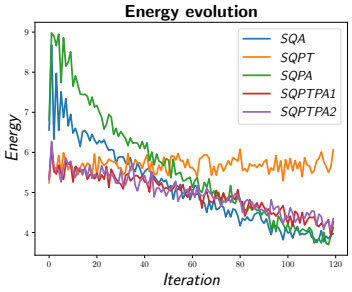
(c) Energy evolution of a 51-variable clustering problem obtained considering 32 total copies and 5100 MC_step



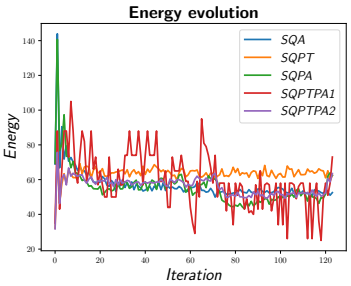
(d) Energy evolution of a 9-pot and 9-plant (81-variable) garden optimization problem obtained considering 32 total copies and 40500 MC_step



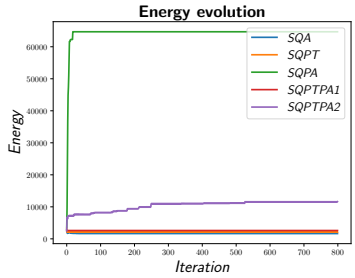
(e) Energy evolution of a 4-nurse and 20-day (80-variable) nurse scheduling problem obtained considering 18 total copies and 800 MC_step



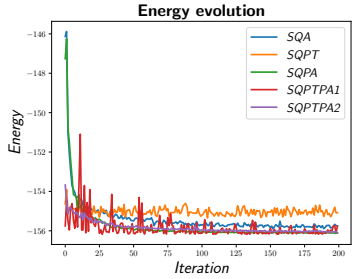
(f) Energy evolution of a 10-node and 3-colour (30-variable) graph colouring problem obtained considering 18 total copies and 120 MC_step



(g) Energy evolution of a 70-node minimum vertex cover problem obtained considering 18 total copies and 280 MC_step



(h) Energy evolution of a 200-node number partitioning problem obtained considering 18 total copies and 800 MC_step



(i) Energy evolution of a 100-power (400-variable) linear regression problem obtained considering 18 total copies and 200 MC_step

FIGURE 17: Energy evolutions obtained by averaging values obtained in each iteration in the on-hundred running each solver for each optimization

values of the features in the clustering one or the numbers in the starting set of number partitioning. The problems size was increased gradually to define a sufficient number of iterations to reach the optimal value (estimated by solving each problem with the **qubovert simulated annealer** on an extremely long annealing duration) by at least one solver and that the majority of the algorithms achieved a value close to the optimal one. Moreover, the gradual increase of the problem dimension permits to clearly identify the considered figures of merit trends and, consequently, to predict them for larger optimizations.

B. FIGURES OF MERIT

The effectiveness of a solver in optimizing a objective function was evaluated in terms of **optimal value** (**opt**), **average value** (**avg**), **probability** (p_{range}) of obtaining a value which is the optimal one or lower than it by a certain percentage (p_{cons}) and the **time-to-solution** (**TTS**).

In particular, the optimal value is the lowest final value obtained by each solver in the one hundred runs, while the average one is obtained by averaging the one hundred obtained final values. These metrics are the most intuitive to take into account. In fact, lower best-obtained and average energy values imply that the results provided by the solver have higher quality.

In addition, the p_{range} was evaluated. This is the **probability of obtaining final energy lower than**:

$$\text{val} = \text{opt} + \left| \text{opt} \cdot p_{\text{cons}} \right| , \tag{70}$$

where opt is the expected optimal value. It is possible to say that one solver is better than another if its p_{range} is higher. This metric is defined as:

$$p_{\text{range}} \triangleq \frac{n_{\text{in\_range}}}{n_{\text{tot}}} 100 , \tag{71}$$

where $n_{\text{in\_range}}$ is the number of times in which the solver achieved final energy lower than val, and n_{tot} is the number of runs. In order to appreciate the meaning of this metric, it is important to remind that p_{range} is expected to depend on the number of Monte Carlo steps; in particular, a higher number of steps will increase p_{range}. For this reason, the *normalized* $\frac{p_{\text{range}}}{\text{MC\_step}}$, varying the problem size for each type of considered optimization problem and for each solver, is reported (Figure 14).

As shown in Figure 13a, the **moving average** (**MA**) of the obtained samples with a window size w was applied to smooth the variation and to identify easier the trend:

$$\text{MA} = \frac{1}{w} \sum_{i=0}^{w-1} x[k-i] , \tag{72}$$

where x is the array of data to be moving-averaged and $k \geq 1$ the index of the last sample of the window. For $k < w$, the first element of x is replicated $w - k$ times.

The most complex and complete metric considered is the **time-to-solution** (TTS). This figure of merit is commonly employed in the literature for comparing heuristic algorithms and in particular, for detecting quantum speed-up [96]–[100] given by quantum, quantum-inspired and quantum-compliant optimization approaches. It is defined as the time required to find a **target solution**, which is the optimal one or a sub-optimal one with final energy lower than a certain value with a percentage of confidence p_{conf}, usually set to 99%. In particular, it can be computed as:

$$\text{TTS} = t_f \frac{\log\left(1 - p_{\text{conf}}\right)}{\log\left(1 - p_{\text{range}}(t_f)\right)} , \tag{73}$$

where t_f is the algorithm execution time, $p_{\text{range}}(t_f))$ is the probability of finding energy lower than a certain value, executing the algorithm for a time t_f. In this work, considering that the implementations were realized in Python language without any particular code optimization, the time t_f and, consequently, the TTS are expressed in terms of the number of iterations. Indeed, as explained in Section III, the iterations of the analyzed algorithm and execution time can be reasonably considered equivalent.

In the evaluation of this metric, some particular cases were managed:

- if p_{range} is equal to 0, TTS was computed by considering a p_{range} equal to 0.1%;
- if p_{range} is equal to 100%, TTS was computed by considering a p_{range} equal to 99%.

This parameter is expected to grow exponentially with the problem dimension [97] for QA and for SQA:

$$\text{TTS} \approx 10^{a + b\sqrt{n} + c \log\left(\sqrt{n}\right)} , \tag{74}$$

where n is the number of involved binary variables, a, b and c are interpolation coefficients. In order to verify this, the evolution of TTS varying the number of binary variables involved is reported in logarithmic scale for each type of considered optimization problem and for each solver (Figure 15). As shown in Figure 13b, the **moving average** (**MA**) of the obtained samples with a window size w was applied to smooth the variation and to identify more easily the trend.

In addition to these explicit figures of merit, the energy evolution (Figure 17) and the cumulative distribution (Figure 16) — obtained through the multiple repetitions of the algorithm — of one of the solved problems for each optimization category are reported.

C. PERFORMED TESTS

This paragraph reports and comments on the obtained results for each kind of **benchmark** problem. In particular, three tables for each are shown: one with the setup of each problem, one for the number of Monte Carlo steps, opt and avg and one with p_{range} and TTS.

1) Max-cut

Table 1 reports the setup configuration of the performed max-cut tests, in particular the number of Trotters, system copies and binary variables. Each problem is identified with a name

TABLE 1: Setup configuration considered for solving one hundred times max-cut problems of different sizes with the proposed algorithms and the SQA as a reference. The initial values for transverse fields and temperature are both equal to one for all the tests, while the number of involved copies differs according to the problem and is reported in the following.

Problem		$N_{\text{tot\_copies}}$	**SQA**	**SQPT**		**SQPA**		**SQPTPA1**			**SQPTPA2**		
Name	v_q		M	M	SYS	M	SYS	M	SYS_{temp}	SYS_{pop}	M	SYS_{temp}	SYS_{pop}
MaxCut_100	100	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_105	105	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_110	110	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_115	115	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_120	120	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_135	135	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_140	140	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_145	145	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_150	150	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_155	155	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_160	160	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_165	165	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_170	170	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_175	175	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_180	180	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_185	185	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_190	190	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_195	195	18	18	3	6	3	6	3	3	3	3	3	3
MaxCut_200	200	18	18	3	6	3	6	3	3	3	3	3	3

TABLE 2: Results obtained by solving one hundred times max-cut problems of different sizes with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 1. In this table, the energy of the best solution between ones obtained by each solver in the one hundred repetitions is reported (**opt**) together with the average of the final energies found.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	**MC_step**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**
MaxCut_100	400	-13667.00	-13662.26	-13667.00	-13643.57	-13667.00	-13666.45	**-13667.00**	**-13666.53**	-13667.00	-13666.20
MaxCut_105	420	-14934.00	-14926.41	-14934.00	-14908.19	-14934.00	-14931.61	**-14934.00**	**-14931.87**	-14934.00	-14930.71
MaxCut_110	440	-16747.00	-16737.59	-16747.00	-16713.41	**-16747.00**	**-16743.47**	-16747.00	-16742.53	-16747.00	-16742.04
MaxCut_115	460	-17911.00	-17898.38	-17911.00	-17883.37	**-17911.00**	**-17902.88**	-17911.00	-17903.58	-17911.00	-17902.30
MaxCut_120	480	-19471.00	-19460.71	-19471.00	-19442.06	**-19471.00**	**-19466.32**	-19471.00	-19465.20	-19471.00	-19464.41
MaxCut_135	540	-24621.00	-24608.07	-24621.00	-24571.86	**-24621.00**	**-24620.94**	-24621.00	-24620.10	**-24621.00**	**-24620.56**
MaxCut_140	560	-26485.00	-26464.60	-26485.00	-26452.52	-26485.00	-26472.43	**-26485.00**	**-26472.35**	-26485.00	-26471.79
MaxCut_145	580	-28375.00	-28357.20	-28375.00	-28321.42	-28375.00	-28371.07	**-28375.00**	**-28372.52**	-28375.00	-28369.23
MaxCut_150	600	-30101.00	-30086.69	-30101.00	-30047.61	**-30101.00**	**-30094.55**	-30101.00	-30094.60	-30101.00	-30095.31
MaxCut_155	620	-32078.00	-32062.33	-32078.00	-31993.13	**-32078.00**	**-32076.96**	-32078.00	-32075.31	-32078.00	-32076.36
MaxCut_160	640	-34295.00	-34270.65	-34295.00	-34223.67	-34295.00	-34286.62	-34295.00	-34286.86	**-34295.00**	**-34288.28**
MaxCut_165	660	-36135.00	-36117.01	-36135.00	-36078.99	**-36135.00**	**-36131.86**	**-36135.00**	**-36131.58**	-36135.00	-36130.24
MaxCut_170	680	-38574.00	-38554.81	-38574.00	-38510.76	**-38574.00**	**-38570.10**	**-38574.00**	**-38569.27**	-38574.00	-38567.92
MaxCut_175	700	-41002.00	-40948.24	-41002.00	-40924.70	-41002.00	-40972.66	-41002.00	-40976.09	**-41002.00**	**-40977.16**
MaxCut_180	720	-43417.00	-43406.33	-43417.00	-43361.54	**-43417.00**	**-43413.72**	**-43417.00**	**-43414.91**	-43417.00	-43410.70
MaxCut_185	740	-45320.00	-45292.85	-45320.00	-45241.92	-45320.00	-45313.89	**-45320.00**	**-45317.23**	-45320.00	-45313.94
MaxCut_190	760	-48060.00	-48025.74	-48060.00	-47979.10	-48060.00	-48047.50	**-48060.00**	**-48046.80**	-48060.00	-48046.85
MaxCut_195	780	-50872.00	-50853.35	-50872.00	-50793.23	**-50872.00**	**-50869.77**	**-50872.00**	**-50868.34**	**-50872.00**	**-50866.92**
MaxCut_200	800	-53286.00	-53254.13	-53286.00	-53201.93	**-53286.00**	**-53278.44**	**-53286.00**	**-53278.74**	**-53286.00**	**-53277.01**

TABLE 3: Results obtained by solving one hundred times max-cut problems of different sizes with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 1. In this table, the probability of finding the final energy in a given range (p_{range}) and the TTS.

Problem		SQA		SQPT		SQPA		SQPTPA1		SQPTPA2	
Name	$p_{\text{cons}}[\%]$	$p_{\text{range}}[\%]$	TTS	$p_{\text{range}}[\%]$	TTS	$p_{\text{range}}[\%]$	TTS	$p_{\text{range}}[\%]$	TTS	$p_{\text{range}}[\%]$	TTS
MaxCut_100	0.10	90.00	800.00	49.00	2735.70	98.00	470.87	**99.00**	**400.00**	98.00	470.87
MaxCut_105	0.10	77.00	1316.05	42.00	3550.72	99.00	420.00	**100.00**	**420.00**	98.00	494.42
MaxCut_110	0.10	84.00	1105.69	51.00	2840.51	**94.00**	**720.22**	91.00	841.50	93.00	761.97
MaxCut_115	0.10	74.00	1572.58	47.00	3336.67	**95.00**	**707.13**	94.00	752.96	93.00	796.60
MaxCut_120	0.10	77.00	1504.06	24.00	8054.61	**95.00**	**737.88**	94.00	785.69	94.00	785.69
MaxCut_135	0.10	86.00	1264.83	56.00	3029.05	**100.00**	**540.00**	99.00	540.00	**100.00**	**540.00**
MaxCut_140	0.10	75.00	1860.28	51.00	3615.19	98.00	659.22	**99.00**	**560.00**	97.00	735.45
MaxCut_145	0.10	72.00	2098.25	28.00	8130.79	97.00	761.72	**98.00**	**682.77**	95.00	891.60
MaxCut_150	0.10	86.00	1405.36	26.00	9176.54	**98.00**	**706.31**	97.00	787.98	97.00	787.98
MaxCut_155	0.10	74.00	2119.56	20.00	12795.38	**98.00**	**729.85**	94.00	1014.86	96.00	887.02
MaxCut_160	0.10	62.00	3046.05	16.00	16904.23	87.00	1444.60	85.00	1553.57	**89.00**	**1335.27**
MaxCut_165	0.10	88.00	1433.51	52.00	4141.06	**100.00**	**660.00**	**100.00**	**660.00**	99.00	660.00
MaxCut_170	0.10	85.00	1650.67	46.00	5082.09	**100.00**	**680.00**	**100.00**	**680.00**	99.00	680.00
MaxCut_175	0.10	29.00	9412.29	18.00	16243.91	70.00	2677.49	71.00	2604.16	**79.00**	**2065.56**
MaxCut_180	0.10	94.00	1178.54	47.00	5222.61	**100.00**	**720.00**	**100.00**	**720.00**	99.00	720.00
MaxCut_185	0.10	75.00	2458.23	38.00	7128.81	92.00	1349.24	**98.00**	**871.12**	96.00	1058.70
MaxCut_190	0.10	74.00	2598.17	36.00	7842.33	95.00	1168.31	**96.00**	**1087.31**	93.00	1316.13
MaxCut_195	0.10	90.00	1560.00	38.00	7514.15	**100.00**	**780.00**	**100.00**	**780.00**	**100.00**	**780.00**
MaxCut_200	0.10	74.00	2734.92	30.00	10329.11	**97.00**	**1050.64**	**97.00**	**1050.64**	**97.00**	**1050.64**

in the format: `MaxCut_`n_n, where n_n is the number of nodes. In this case, the $N_{\text{tot\_copies}}$ is equal to 18 for all the considered problems.

Table 2 shows the optimal value (opt), which is the lowest final value obtained by each solver in the one hundred runs considered, the average one (avg), which is obtained by averaging the one hundred obtained final values, and the number of iterations for obtaining these (MC_step).

Table 3 provides the p_{range}, the TTS and the considered p_{cons}. The latter was fixed at 0.1 % for all the considered problems because all solvers reached the optimal value at least once, and the value obtained is sufficiently close to it to appreciate p_{range} lower than 100% [1].

The $\frac{p_{\text{range}}}{\text{MC\_step}}$ and the TTS evolutions, varying the problem size with a window w equal to 50, are reported in Figures 14a and 15a. The obtained TTS shape is coherent with expectations. In fact, it is possible to recognize a square-root evolution with the problem size in the logarithmic axis ($10^{a+b\sqrt{n}+c\log(\sqrt{n})} \approx 10^{b\sqrt{n}}$, for $n \to \infty$ and $a \simeq b \simeq c$).

It is possible to notice that SQPTPA1 and SQPA provide the best results, while SQPT has the worst ones. Indeed, the highest p_{range} and the lowest TTS were found with the first solvers. At the same time, SQPT provides a TTS significantly higher than the other solvers. Instead, the SQA performance is better than SQPT but significantly worse than SQPTPA1 and SQPA, while SQPTPA2 results quality is close to SQPTPA1 and SQPA.

Examples of cumulative distribution and energy evolutions are reported in Figures 16a and 17a, respectively. It is possible to observe that the cumulative distributions of SQPT, SQPTPA1 and SQPTPA2 are more concentrated on the left, i.e. the probability of reaching the optimal solution is higher, as explained in Figure 13c. At the same time, it is possible to observe that SQPTA1 has a faster convergence to the optimal than the other algorithms.

Considering all the analyzed figures of merit, it is possible to conclude that the **SQPTPA1 algorithm is the most suitable for exploring the max-cut problems energy profile**.

2) Clustering

Table 4 reports the setup configuration of the performed clustering tests, in particular the number of Trotters, system copies and binary variables. Each problem is identified with a name in the format: `Clustering_`x, where x is the number of considered data (the number of clusters was fixed to four and, consequently, the number of involved binary variables is equal to $4x$). In this case, the $N_{\text{tot\_copies}}$ is equal to 18 for the smaller problem considered and equal to 32 for all the others. Due to the rapid growth of the problem size increasing the number of data, the number of performed tests is lower than in the max-cut case.

Table 5 shows the difference between the actual optimal value and the lowest final value obtained by each solver in the one hundred runs (Δopt), one between the actual optimal value and the average one (Δavg), which is obtained by averaging the one hundred obtained final values, and the number of iterations for obtaining these (MC_step). In this case, the choice of showing the divergence from the

[1] A higher p_{range} could imply a $p_{\text{range}} = 100\%$, thus making difficult the comparison of the quality of the solutions provided by different solvers.

TABLE 4: Setup configuration considered for solving one hundred times clustering problems (`Clustering_`x) of different sizes (four-cluster, x-data) with the proposed algorithms and the SQA as a reference. The initial values for transverse fields and temperature are both equal to one for all the tests, while the number of involved copies differs according to the problem and is reported in the following.

Problem		$N_{\text{tot\_copies}}$	**SQA**	**SQPT**		**SQPA**		**SQPTPA1**			**SQPTPA2**		
Name	v_q		M	M	SYS	M	SYS	M	SYS_{temp}	SYS_{pop}	M	SYS_{temp}	SYS_{pop}
Clustering_4	16	18	18	3	6	3	6	3	3	3	3	3	3
Clustering_5	20	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_6	24	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_7	28	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_8	32	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_9	36	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_10	40	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_11	44	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_12	48	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_13	52	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_14	56	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_15	60	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_16	64	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_17	68	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_18	72	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_19	76	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_20	80	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_21	84	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_22	88	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_23	92	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_24	96	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_25	100	32	32	4	8	4	8	4	4	4	4	4	4
Clustering_26	104	32	32	4	8	4	8	4	4	4	4	4	4

actual optimal solution, instead of the actual obtained value, was done because, in this type of optimization, the energy difference among some solutions is so small that is not possible to appreciate it by reading the absolute value without considering an excessive number of decimal digits.

Table 6 provides the p_{range}, the TTS and the considered p_{cons}. The latter is computed for each optimization problem according to the following formula:

$$p_{\text{cons}}[\%] = \left| 1 - \frac{\min_h}{\min_l} \right| 100\,, \tag{75}$$

where $\min_h$ is the highest opt among ones of all the analyzed solvers, while $\min_l$ is the lowest one. In this way, the value obtained (val) is sufficiently close to the actual optimal value to appreciate p_{range} lower than 100% and sufficiently high to obtain a p_{range} higher than 0% in all cases.

The $\frac{p_{\text{range}}}{\text{MC\_step}}$ and the TTS evolutions, varying the problem size with a window w equal to 20, are reported in Figures 14b and 15b. The obtained TTS shape is coherent with expectations. In fact, it is possible to recognize a square-root evolution — even if less evident than in the max-cut case due to the lower number of performed tests — with the problem size in the logarithmic axis.

It is possible to notice that SQPT, SQPTPA1 and SQPTPA2 provide the best results, while SQA has the worst ones. Indeed, the highest p_{range} and the lowest TTS were found with the first solvers. At the same time, SQPA provides TTS significantly higher than the other solvers.

Examples of cumulative distribution and energy evolutions are reported in Figures 16b and 17b respectively. It is possible to observe that the cumulative distributions of SQPT, SQPA, SQPTPA1 and SQPTPA2 are more concentrated on the left, i.e. the probability of reaching the optimal solution is higher, as explained in Figure 13c. At the same time, it is possible to observe that SQPTA1 and SQPTA2 have a faster and better convergence to the optimal than the other algorithms. On the other hand, the SQPA reaches the optimal solution but has a slow convergence, This implies that it does not represent the best exploration approach in this case.

Considering all the analyzed figures of merit, it is possible to conclude that the **SQPTPA1 and SQPTA2 algorithms are the most suitable for exploring the clustering problems energy profile**.

3) Knapsack

Table 7 reports the algorithms configurations exploited for solving knapsack problems, in particular the number of Trotters, system copies and binary variables. Each problem is identified with a name in the format: `Knapsack_`x, where x is the number of involved binary variables. In this case, the $N_{\text{tot\_copies}}$ increases from 10 to 32 with the problem size. For knapsack problems, a granular and linear increase in the problem size is impossible because auxiliary variables are introduced to represent the inequality constraint. Therefore,

TABLE 5: Results obtained by solving one hundred times clustering problems (`Clustering_`x) of different sizes (four-cluster, x-data) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 4. In this table, the difference between the actual optimal energy and ones of the best solution found by each solver in the one hundred repetitions is reported (**Δopt**) together with the difference between the actual optimal energy and the average of the final energies found (**Δavg**).

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	**MC_step**	**Δopt**	**Δavg**	**Δopt**	**Δavg**	**Δopt**	**Δavg**	**Δopt**	**Δavg**	**Δopt**	**Δavg**
Clustering_4	16	0.00e+00	7.03e-13	**0.00e+00**	**5.89e-13**	6.82e-13	9.00e-13	2.27e-13	6.03e-13	0.00e+00	5.84e-13
Clustering_5	200	4.77e-12	7.73e-12	2.73e-12	6.62e-12	8.19e-12	9.51e-12	6.82e-13	5.08e-12	**0.00e+00**	**4.87e-12**
Clustering_6	240	2.27e-12	8.05e-12	**4.55e-13**	**6.46e-12**	7.28e-12	1.03e-11	1.82e-12	6.82e-12	0.00e+00	6.36e-12
Clustering_7	280	3.64e-12	1.46e+00	**0.00e+00**	**4.94e-12**	4.09e-12	9.05e-12	4.55e-13	6.83e-12	4.55e-13	6.38e-12
Clustering_8	640	1.00e-11	1.21e+01	**9.09e-13**	**2.03e-11**	3.87e-11	5.48e-11	0.00e+00	2.08e-11	**4.55e-12**	**1.87e-11**
Clustering_9	720	1.18e-11	3.02e+01	**4.55e-12**	**1.60e-11**	2.73e-12	2.26e-11	0.00e+00	2.23e-11	2.73e-12	2.29e-11
Clustering_10	800	8.09e-11	6.01e+01	**3.18e-11**	**6.36e-11**	7.09e-11	1.02e-10	1.64e-11	6.91e-11	0.00e+00	5.72e-11
Clustering_11	880	1.04e-10	7.82e+01	**3.82e-11**	**9.04e-11**	6.73e-11	1.24e-10	1.64e-11	1.03e-10	0.00e+00	8.71e-11
Clustering_12	960	2.17e+01	1.15e+02	**1.44e-10**	**1.88e-10**	**1.87e-10**	**2.59e-10**	**0.00e+00**	**1.70e-10**	**2.82e-11**	**1.23e-10**
Clustering_13	1040	1.51e-10	1.20e+02	**1.64e-11**	**7.85e-11**	6.37e-11	1.36e-10	0.00e+00	1.08e-10	**5.46e-12**	**9.52e-11**
Clustering_14	1120	1.06e+01	1.40e+02	**8.19e-11**	**1.97e-10**	**2.07e-10**	**2.84e-10**	**3.27e-11**	**2.33e-10**	**0.00e+00**	**1.71e-10**
Clustering_15	1200	1.96e-10	9.37e+01	**5.64e-11**	**1.16e-10**	0.00e+00	1.72e-10	1.09e-11	1.56e-10	6.55e-11	1.48e-10
Clustering_16	1280	3.00e-10	8.33e+01	**9.82e-11**	**1.93e-10**	1.42e-10	2.92e-10	1.07e-10	2.77e-10	**0.00e+00**	**1.62e-10**
Clustering_17	1360	8.66e-10	7.79e+01	**4.69e-10**	**6.51e-10**	7.48e-10	9.16e-10	3.27e-10	8.68e-10	0.00e+00	5.51e-10
Clustering_18	1440	1.17e+00	6.77e+01	**0.00e+00**	**2.25e-10**	**1.22e-10**	**2.80e-10**	**1.55e-10**	**4.10e-04**	**0.00e+00**	**2.05e-04**
Clustering_19	1520	5.61e+00	5.22e+01	**4.40e-10**	**6.47e-01**	**3.35e-10**	**6.12e-01**	**4.44e-10**	**7.48e-01**	**0.00e+00**	**5.46e-01**
Clustering_20	1600	9.00e+00	4.86e+01	**1.82e-10**	**3.62e+00**	3.38e-10	3.12e+00	3.67e-10	6.10e+00	0.00e+00	3.57e+00
Clustering_21	1680	9.58e+00	5.26e+01	**8.22e-10**	**9.58e-01**	**3.67e-10**	**1.21e-01**	**6.33e-10**	**4.23e-01**	**0.00e+00**	**1.09e-01**
Clustering_22	1760	5.27e+00	2.45e+01	2.33e-10	3.31e+00	**1.09e-10**	**2.49e+00**	2.29e-10	3.42e+00	0.00e+00	3.49e+00
Clustering_23	1840	2.21e+00	1.28e+01	1.82e-09	7.46e-01	**1.38e-10**	**2.66e-01**	0.00e+00	8.86e-01	4.37e-10	5.05e-01
Clustering_24	1920	1.31e+00	2.28e+01	**1.82e-11**	**3.15e+00**	1.78e-10	6.43e+00	0.00e+00	1.10e+01	1.56e-10	9.45e+00
Clustering_25	2000	2.12e+00	3.38e+01	4.15e-10	1.23e+01	**5.28e-10**	**3.32e+00**	5.68e-10	5.23e+00	0.00e+00	4.82e+00
Clustering_26	2080	1.31e+01	6.62e+01	**8.15e-10**	**1.02e+00**	1.96e-10	7.04e+00	0.00e+00	7.61e+00	3.20e-10	6.76e+00

TABLE 6: Results obtained by solving one hundred times clustering problems (`Clustering_`x) of different sizes (four-cluster, x-data) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 4. In this table, the probability of finding the final energy in a given range (p_{range}) and the TTS.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	$p_{\text{cons}}[\%]$	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**
Clustering_4	7.50e+01	58.00	84.94	**88.00**	**34.75**	13.00	529.09	82.00	42.97	79.00	47.21
Clustering_5	1.24e-10	74.00	683.73	99.00	200.00	5.00	17956.23	99.00	200.00	**100.00**	**200.00**
Clustering_6	3.56e-11	34.00	2659.93	**76.00**	**774.46**	4.00	27074.64	69.00	943.70	67.00	996.91
Clustering_7	1.29e-11	3.00	42333.59	**35.00**	**2993.27**	2.00	63825.49	10.00	12238.43	16.00	7395.60
Clustering_8	7.17e-11	20.00	13208.13	**100.00**	**640.00**	1.00	293254.77	95.00	983.84	**100.00**	**640.00**
Clustering_9	1.50e-11	2.00	164122.68	**14.00**	**21984.21**	13.00	23809.23	6.00	53587.09	5.00	64642.42
Clustering_10	1.54e-10	1.00	366568.46	**99.00**	**800.00**	6.00	59541.21	73.00	2813.75	90.00	1600.00
Clustering_11	9.29e-11	1.00	403225.31	**89.00**	**1836.00**	8.00	48602.44	41.00	7680.63	73.00	3095.12
Clustering_12	1.67e+01	1.00	439882.15	**100.00**	**960.00**	**100.00**	**960.00**	**100.00**	**960.00**	**100.00**	**960.00**
Clustering_13	1.10e-10	1.00	476539.00	**100.00**	**1040.00**	77.00	3258.80	99.00	1040.00	**100.00**	**1040.00**
Clustering_14	5.13e+00	1.00	513195.85	**100.00**	**1120.00**	**100.00**	**1120.00**	**100.00**	**1120.00**	**100.00**	**1120.00**
Clustering_15	7.79e-11	1.00	549852.69	**100.00**	**1200.00**	73.00	4220.62	91.00	2294.99	95.00	1844.69
Clustering_16	1.03e-10	1.00	586509.54	**100.00**	**1280.00**	52.00	8031.15	63.00	5928.69	**100.00**	**1280.00**
Clustering_17	3.08e-10	1.00	623166.38	**100.00**	**1360.00**	17.00	33612.65	35.00	14538.72	99.00	1360.00
Clustering_18	3.26e-01	1.00	659823.23	**100.00**	**1440.00**	**100.00**	**1440.00**	**100.00**	**1440.00**	**100.00**	**1440.00**
Clustering_19	1.42e+00	1.00	696480.08	**100.00**	**1520.00**	**100.00**	**1520.00**	**100.00**	**1520.00**	**100.00**	**1520.00**
Clustering_20	2.26e+00	1.00	733136.92	**89.00**	**3338.18**	88.00	3475.17	72.00	5788.27	86.00	3747.63
Clustering_21	2.04e+00	1.00	769793.77	**100.00**	**1680.00**	**100.00**	**1680.00**	**100.00**	**1680.00**	**100.00**	**1680.00**
Clustering_22	1.02e+00	1.00	806450.61	76.00	5679.35	**94.00**	**2880.88**	81.00	4880.44	84.00	4422.78
Clustering_23	4.64e-01	1.00	843107.46	86.00	4309.78	**96.00**	**2632.44**	89.00	3838.90	90.00	3680.00
Clustering_24	2.57e-01	1.00	879764.31	**44.00**	**15249.47**	25.00	30735.06	14.00	58624.57	13.00	63491.28
Clustering_25	3.58e-01	1.00	916421.15	11.00	79035.77	**39.00**	**18633.24**	23.00	35239.41	23.00	35239.41
Clustering_26	1.88e+00	1.00	953078.00	**99.00**	**2080.00**	65.00	9124.17	63.00	9634.13	71.00	7738.07

TABLE 7: Setup configuration considered for solving one hundred times knapsack problems (`Knapsack_`x) of different sizes (x-variable) with the proposed algorithms and the SQA as a reference. The initial values for transverse fields and temperature are both equal to one for all the tests, while the number of involved copies differs according to the problem and is reported in the following.

Problem		$N_{\text{tot\_copies}}$	**SQA**	**SQPT**		**SQPA**		**SQPTPA1**			**SQPTPA2**		
Name	v_q		M	M	SYS	M	SYS	M	SYS_{temp}	SYS_{pop}	M	SYS_{temp}	SYS_{pop}
Knapsack_10	10	10	10	2	5	2	5	2	3	2	2	3	2
Knapsack_20	20	10	10	2	5	2	5	2	3	2	2	3	2
Knapsack_25	25	10	10	2	5	2	5	2	3	2	2	3	2
Knapsack_30	30	18	18	3	6	3	6	3	3	3	3	3	3
Knapsack_35	35	18	18	3	6	3	6	3	3	3	3	3	3
Knapsack_41	41	18	18	3	6	3	6	3	3	3	3	3	3
Knapsack_46	46	32	32	4	8	4	8	4	4	4	4	4	4
Knapsack_51	51	32	32	4	8	4	8	4	4	4	4	4	4
Knapsack_52	52	32	32	4	8	4	8	4	4	4	4	4	4
Knapsack_53	53	32	32	4	8	3	6	3	3	3	3	3	3
Knapsack_56	56	32	32	4	8	4	8	4	4	4	4	4	4
Knapsack_57	57	32	32	4	8	4	8	4	4	4	4	4	4
Knapsack_58	58	32	32	4	8	4	8	4	4	4	4	4	4
Knapsack_59	59	32	32	4	8	4	8	4	4	4	4	4	4

TABLE 8: Results obtained by solving one hundred times knapsack problems (`Knapsack_`x) of different sizes (x-variable) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 7. In this table, the energy of the best solution between ones obtained by each solver in the one hundred repetitions is reported (**opt**) together with the average of the final energies found.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	**MC_step**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**
Knapsack_10	10	**-12.80**	**-12.20**	-12.80	-11.81	-12.80	-11.57	-12.80	-11.89	-12.80	-12.04
Knapsack_20	200	-15.00	-14.15	-15.00	-13.15	**-15.00**	**-14.59**	-15.00	-14.42	-15.00	-14.41
Knapsack_25	250	-15.80	-11.85	-14.80	-10.44	**-15.80**	**-14.45**	-15.80	-13.92	-15.80	-13.73
Knapsack_30	3000	-17.00	-14.24	-16.80	-12.73	**-17.00**	**-16.60**	-17.00	-16.10	-17.00	-16.22
Knapsack_35	3500	-14.80	-12.42	-13.00	-10.37	**-14.80**	**-14.00**	-14.80	-13.79	-14.80	-13.70
Knapsack_41	4100	-21.80	-17.36	-22.20	-15.55	**-23.80**	**-21.93**	-23.80	-21.48	-23.80	-21.54
Knapsack_46	4600	-22.80	-16.19	-22.80	-21.60	-20.80	-17.19	**-22.80**	**-21.91**	**-22.80**	**-21.96**
Knapsack_51	5100	-19.20	-15.30	**-21.80**	**-20.68**	-20.00	-15.98	**-21.80**	**-20.95**	**-21.80**	**-20.84**
Knapsack_52	5200	-21.00	-16.77	**-23.80**	**-22.13**	-19.20	-16.21	**-23.80**	**-22.33**	**-23.80**	**-22.31**
Knapsack_53	5300	-17.80	-10.41	**-18.80**	**-17.70**	-18.80	-16.98	-18.80	-17.22	-18.80	-17.54
Knapsack_54	5400	-14.80	1.86	**-16.80**	**-15.87**	-16.80	-14.82	-15.80	-14.99	-16.80	-15.07
Knapsack_55	5500	-18.80	-9.15	**-24.80**	**-22.31**	-22.20	-19.51	**-23.80**	**-21.17**	-24.20	-21.39
Knapsack_56	16800	-17.80	-1.18	**-18.80**	**-17.58**	-16.20	-10.03	**-18.80**	**-17.60**	**-18.80**	**-17.67**
Knapsack_57	17100	-19.80	-10.07	**-22.80**	**-21.37**	-19.20	-15.17	**-22.80**	**-21.70**	**-22.80**	**-21.76**
Knapsack_58	17400	-16.80	-8.26	**-18.80**	**-18.13**	-16.80	-13.69	**-18.80**	**-17.89**	**-18.80**	**-18.03**
Knapsack_59	17700	-19.00	-10.82	**-23.80**	**-22.50**	-19.20	-14.54	**-23.80**	**-21.81**	**-23.80**	**-22.41**

the number of performed tests is lower than in the max-cut case since the problem grows very fast.

Table 8 shows the optimal value (opt), the average one (avg), and the number of iterations for obtaining these (MC_step). Table 9 provides the p_{range}, the TTS and the considered p_{cons}. In case of achievement of the optimal solution by all solvers, the latter is fixed to 0.01%, while if even just one solver does not reach the expected energy, it is computed for each optimization problem according to the formula reported in the equation 75. In this way, the value obtained (val) is sufficiently close to the actual optimal value to appreciate p_{range} lower than 100% and sufficiently high to obtain a p_{range} higher than 0% in all cases.

The $\frac{p_{\text{range}}}{\text{MC\_step}}$ and the TTS evolutions, varying the problem size with a window w equal to 20, are reported in Figures 14c and 15c. Also in this case, the obtained TTS shape is coherent with expectation. In fact, it is possible to recognize, despite the limited number of samples, a square-root evolution with the problem size in the logarithmic axis.

Observing the obtained results, it is possible to say that the best solver for small size problems (until 41-variable one) is the SQPA — since it has the highest p_{range} and the lowest TTS —, while for larger ones the best performance are provided by SQPTPA1 and SQPTPA2. The SQA solver gives the worst results — i.e. TTS significantly higher than the other solvers — for all the considered problem sizes. Examples of

TABLE 9: Results obtained by solving one hundred times knapsack problems (`Knapsack_`x) of different sizes (x-variable) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 7. In this table, the probability of finding the final energy in a given range (p_{range}) and the TTS.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	$p_{\text{cons}}[\%]$	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**
Knapsack_10	0.01	**39.00**	**93.17**	24.00	167.80	14.00	305.34	24.00	167.80	28.00	140.19
Knapsack_20	0.01	18.00	4641.12	4.00	22562.20	**42.00**	**1690.82**	25.00	3201.57	27.00	2926.60
Knapsack_25	6.33	8.00	13807.51	1.00	114552.64	**52.00**	**1568.58**	31.00	3102.68	20.00	5159.43
Knapsack_30	1.18	4.00	338433.01	1.00	1374631.73	**35.00**	**32070.70**	13.00	99205.12	24.00	50341.31
Knapsack_35	12.16	19.00	76490.21	1.00	1603737.02	**96.00**	**5007.37**	81.00	9705.42	80.00	10014.74
Knapsack_41	8.40	1.00	1878663.36	2.00	934587.50	**55.00**	**23645.61**	34.00	45440.42	46.00	30642.04
Knapsack_46	8.77	1.00	2107768.65	92.00	8387.20	1.00	2107768.65	**100.00**	**4600.00**	**100.00**	**4600.00**
Knapsack_51	11.93	1.00	2336873.94	**100.00**	**5100.00**	1.00	2336873.94	**100.00**	**5100.00**	**100.00**	**5100.00**
Knapsack_52	19.33	7.00	329980.11	**100.00**	**5200.00**	1.00	2382695.00	**100.00**	**5200.00**	**100.00**	**5200.00**
Knapsack_53	5.32	1.00	2428516.06	**36.00**	**54689.91**	9.00	258797.86	15.00	150181.90	22.00	98234.20
Knapsack_54	11.90	1.00	2474337.11	**100.00**	**5400.00**	51.00	34860.76	77.00	16920.68	90.00	10800.00
Knapsack_55	24.19	1.00	2520158.17	**100.00**	**5500.00**	70.00	21037.38	**100.00**	**5500.00**	99.00	5500.00
Knapsack_56	13.83	1.00	7697937.69	**100.00**	**16800.00**	1.00	7697937.69	**100.00**	**16800.00**	**100.00**	**16800.00**
Knapsack_57	15.79	1.00	7835400.86	**100.00**	**17100.00**	1.00	7835400.86	**100.00**	**17100.00**	**100.00**	**17100.00**
Knapsack_58	10.64	1.00	7972864.03	**100.00**	**17400.00**	1.00	7972864.03	**100.00**	**17400.00**	**100.00**	**17400.00**
Knapsack_59	20.17	1.00	8110327.20	**100.00**	**17700.00**	1.00	8110327.20	**100.00**	**17700.00**	**100.00**	**17700.00**

TABLE 10: Setup configuration considered for solving one hundred times garden optimization problems (`Garden_`c`_`r) of different sizes (c-column and r-row, i.e. $(c \cdot r)^2$-variable) with the proposed algorithms and the SQA as a reference. The initial values for transverse fields and temperature are both equal to one for all the tests, while the number of involved copies differs according to the problem and is reported in the following.

Problem		$N_{\text{tot\_copies}}$	**SQA**	**SQPT**		**SQPA**		**SQPTPA1**			**SQPTPA2**		
Name	v_q		M	M	SYS	M	SYS	M	SYS_{temp}	SYS_{pop}	M	SYS_{temp}	SYS_{pop}
Garden_1_2	4	10	10	2	5	2	5	2	3	2	2	3	2
Garden_1_3	9	10	10	2	5	2	5	2	3	2	2	3	2
Garden_1_4	16	10	10	2	5	2	5	2	3	2	2	3	2
Garden_2_2	16	10	10	2	5	2	5	2	3	2	2	3	2
Garden_2_3	36	18	18	3	6	3	6	3	3	3	3	3	3
Garden_2_4	64	18	18	3	6	3	6	3	3	3	3	3	3
Garden_3_2	36	18	18	3	6	3	6	3	3	3	3	3	3
Garden_3_3	81	32	32	4	8	4	8	4	4	4	4	4	4

TABLE 11: Results obtained by solving one hundred times garden optimization problems (`Garden_`c`_`r) of different sizes (c-column and r-row, i.e. $(c \cdot r)^2$-variable) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 10. In this table, the energy of the best solution between ones obtained by each solver in the one hundred repetitions is reported (**opt**) together with the average of the final energies found.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	**MC_step**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**
Garden_1_2	2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Garden_1_3	9	**7.00**	**7.00**	**7.00**	**7.00**	7.00	7.10	**7.00**	**7.00**	**7.00**	**7.00**
Garden_1_4	16	0.00	4.35	0.00	0.68	0.00	0.32	**0.00**	**0.21**	**0.00**	**0.21**
Garden_2_2	8	4.00	13.89	4.00	6.34	4.00	4.70	**4.00**	**4.29**	4.00	4.49
Garden_2_3	144	11.00	30.76	10.00	27.13	10.00	22.29	10.00	22.45	**10.00**	**22.06**
Garden_2_4	19200	26.00	48.52	17.00	50.22	**4.00**	**20.36**	6.00	22.86	6.00	23.27
Garden_3_2	10800	3.00	9.24	3.00	8.24	**3.00**	**3.09**	3.00	3.24	3.00	3.23
Garden_3_3	40500	234.00	347.99	**16.00**	**21.83**	34.00	48.50	16.00	24.22	16.00	24.92

TABLE 12: Results obtained by solving one hundred times garden optimization problems (`Garden_c_r`) of different sizes (c-column and r-row, i.e. $(c \cdot r)^2$-variable) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 10. In this table, the probability of finding the final energy in a given range (p_{range}) and the TTS.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	$p_{\text{cons}}[\%]$	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**
Garden_1_2	0.10	**100.00**	**2.00**	**100.00**	**2.00**	**100.00**	**2.00**	**100.00**	**2.00**	**100.00**	**2.00**
Garden_1_3	0.10	**100.00**	**9.00**	**100.00**	**9.00**	99.00	9.00	**100.00**	**9.00**	**100.00**	**9.00**
Garden_1_4	0.10	29.00	215.14	52.00	100.39	68.00	64.67	**79.00**	**47.21**	**79.00**	**47.21**
Garden_2_2	0.10	29.00	107.57	78.00	24.33	93.00	13.85	**98.00**	**9.42**	96.00	11.45
Garden_2_3	10.00	4.00	16244.78	7.00	9137.91	14.00	4396.84	20.00	2971.83	**24.00**	**2416.38**
Garden_2_4	20.00	0.00	88375050.57	0.00	88375050.57	**1.00**	**8797643.07**	0.00	88375050.57	0.00	88375050.57
Garden_3_2	0.10	8.00	596484.51	13.00	357138.44	**91.00**	**20654.88**	77.00	33841.36	78.00	32847.85
Garden_3_3	20.00	0.00	186416122.29	**60.00**	**203548.27**	0.00	186416122.29	43.00	331797.04	39.00	377323.04

TABLE 13: Setup configuration considered for solving one hundred times nurse scheduling problems (`Nurse_n_d`) of different sizes (n-nurse and d-day, i.e. $(n \cdot d)$-variable) with the proposed algorithms and the SQA as a reference. The initial values for transverse fields and temperature are both equal to one for all the tests, while the number of involved copies differs according to the problem and is reported in the following.

Problem		$N_{\text{tot\_copies}}$	**SQA**	**SQPT**		**SQPA**		**SQPTPA1**			**SQPTPA2**		
Name	v_q		M	M	SYS	M	SYS	M	SYS_{temp}	SYS_{pop}	M	SYS_{temp}	SYS_{pop}
Nurse_2_10	20	10	10	2	5	2	5	2	3	2	2	3	2
Nurse_2_15	30	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_2_20	40	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_3_10	30	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_3_15	45	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_3_20	60	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_4_10	40	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_4_15	60	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_4_20	80	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_5_10	50	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_10_10	100	32	32	4	8	4	8	4	4	4	3	3	3
Nurse_10_11	110	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_10_12	120	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_10_13	130	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_10_14	140	18	18	3	6	3	6	3	3	3	3	3	3
Nurse_10_15	150	18	18	3	6	3	6	3	3	3	3	3	3

cumulative distribution and energy evolutions for a large-size problem are reported in Figures 16c and 17c, respectively. It is possible to observe that the cumulative distributions of SQPT, SQPTPA1 and SQPTPA2 are more concentrated on the left, i.e. the probability of reaching the optimal solution is higher. From the energy evolution point of view, it is possible to observe that the SQA reaches the convergence but at an energy value that is higher than the optimal one. Instead, the SQPA reaches a very low value in the middle, but after that starts again to increase the energy without reaching an effective convergence.

Considering all the analyzed figures of merit, it is possible to conclude that the **SQPTPA1 and SQPTPA2 algorithms are the most suitable for exploring the knapsack problems energy profile.**

4) Garden optimization

Table 10 reports the setup configuration of the performed garden optimization tests, in particular the number of Trotters, system copies and binary variables. Each problem is identified with a name in the format: `Garden_c_r`, where c is the number of columns and r is the number of rows, i.e. the number of pots and plants is equal to $c \cdot r$ and, consequently the total number of involved variables is equal to $(c \cdot r)^2$. In this case, the $N_{\text{tot\_copies}}$ increases from 10 to 32 with the problem size. Even this type of problem grows very fast, so it is not possible to do a granular increase of the problem dimension and the number of performed tests is limited.

Table 11 shows the optimal value (opt), the average one (avg), and the number of iterations for obtaining these (MC_step).

Table 12 provides the p_{range}, the TTS and the considered p_{cons}. In case of achievement of the optimal solution by all solvers, the latter is fixed to 0.01%, while if even just one solver does not reach the expected energy, it is computed, exploiting

TABLE 14: Results obtained by solving one hundred times nurse scheduling problems (`Nurse_`n`_`d) of different sizes (n-nurse and d-day, i.e. $(n \cdot d)$-variable) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 13. In this table, the energy of the best solution between ones obtained by each solver in the one hundred repetitions is reported (**opt**) together with the average of the final energies found.

Problem		SQA		SQPT		SQPA		SQPTPA1		SQPTPA2	
Name	MC_step	opt	avg	opt	avg	opt	avg	opt	avg	opt	avg
Nurse_2_10	10	3.00	3.64	**3.00**	**3.34**	3.00	5.06	3.00	3.48	3.00	3.48
Nurse_2_15	300	**-24.00**	**-24.00**	**-24.00**	**-24.00**	**-24.00**	**-24.00**	**-24.00**	**-24.00**	**-24.00**	**-24.00**
Nurse_2_20	400	**-32.00**	**-32.00**	**-32.00**	**-32.00**	**-32.00**	**-32.00**	**-32.00**	**-32.00**	**-32.00**	**-32.00**
Nurse_3_10	300	**6.20**	**6.20**	**6.20**	**6.20**	**6.20**	**6.20**	**6.20**	**6.20**	**6.20**	**6.20**
Nurse_3_15	450	**-94.00**	**-94.00**	-94.00	-93.83	-94.00	-93.82	-94.00	-93.85	-94.00	-93.87
Nurse_3_20	600	**-157.00**	**-157.00**	-157.00	-156.58	-157.00	-156.46	-157.00	-156.62	-157.00	-156.57
Nurse_4_10	400	**7.40**	**7.40**	7.40	7.41	**7.40**	**7.40**	**7.40**	**7.40**	**7.40**	**7.40**
Nurse_4_15	600	-67.00	-66.93	-67.00	-66.99	-67.00	-67.00	-67.00	-66.99	**-67.00**	**-67.00**
Nurse_4_20	800	-125.60	-125.52	-125.60	-125.59	-125.60	-125.57	-125.60	-125.59	**-125.60**	**-125.60**
Nurse_5_4	10	**0.30**	**0.73**	0.30	1.09	0.30	1.61	0.30	1.18	0.30	1.31
Nurse_5_10	500	6.40	6.79	**6.40**	**6.74**	6.00	6.63	6.00	6.65	6.00	6.65
Nurse_10_10	100000	21.70	44.42	**3.00**	**3.92**	6.60	14.36	8.80	14.16	8.20	14.04
Nurse_10_11	1100	11.10	19.73	**5.30**	**17.01**	9.50	16.68	9.30	16.34	8.70	15.86
Nurse_10_12	1200	11.60	23.60	9.40	18.92	**8.00**	**19.74**	8.80	18.18	**7.00**	**18.15**
Nurse_10_13	1300	16.10	27.81	10.50	22.55	12.50	23.02	10.30	20.80	**10.50**	**20.84**
Nurse_10_14	1400	19.70	31.24	**4.00**	**22.97**	12.80	26.52	11.00	22.79	11.20	22.94
Nurse_10_15	1500	23.60	37.03	**14.90**	**27.89**	17.70	29.29	15.90	27.81	**12.70**	**27.62**

also a saturation mechanism, for each optimization problem according to the following formula:

$$p_{\text{cons}}[\%] = \min\left[\left|1 - \frac{\min_h}{\min_l}\right| 100, 20\right], \qquad (76)$$

where $\min_h$ is the highest opt among ones of all the analyzed solvers, while $\min_l$ is the lowest one. In this way, the value obtained (val) is sufficiently close to the actual optimal value to appreciate p_{range} lower than 100%. When the saturation mechanism is employed, some p_{range} are equal to 0%. This choice was done for guaranteeing a val which cannot cause too many p_{range} equal to 100%.

The $\frac{p_{\text{range}}}{\text{MC\_step}}$ and the TTS evolutions, varying the problem size (values obtained with problems of the same size were averaged) with a window w equal to 3, are reported in Figures 14d and 15d. The obtained TTS shape is consistent with expectations. In fact, even if the number of samples is not particularly high, it is possible to recognize a square-root evolution with the problem size in the logarithmic axis.

Observing the obtained results, it is possible to say that the best solvers are the SQPT, the SQPTPA1 and the SQPTPA2, i.e. they provide the highest p_{range} and the lowest TTS. The SQPA assures good results for small-size problems (until 64), while the largest one has not reached the convergence to the optimal value. At the same time, the SQA solver gives the worst results — i.e. TTS significantly higher than the other solvers — for all the considered problem sizes. Examples of cumulative distribution and energy evolutions for the 81-variable problem are reported in Figures 16d and 17d, respectively. It is possible to observe that the cumulative distributions of SQPT, SQPTPA1 and SQPTPA2 are more concentrated on the left, i.e. the probability of reaching the optimal solution is higher. From the energy evolution point of view, it is possible to observe that the SQA reaches the convergence but at an energy value that is higher than the optimal one. Instead, the SQPA reach a very low value in the middle, but after that starts again to increase the energy without reaching an effective convergence.

Considering all the analyzed figures of merit, it is possible to conclude that the **SQPT, SQPTPA1 and SQPTPA2 algorithms are the most suitable for exploring the garden optimization problems objective function.**

5) Nurse scheduling

Table 13 reports the algorithms configurations considered for solving nurse scheduling problems. In particular, the number of Trotters, system copies and binary variables are shown. Each problem is identified with a name in the format: `Nurse_`n`_`d, where n is the number of nurses and d is the number of days, and, consequently, the total number of involved variables is equal to $(n \cdot d)$. In this case, the $N_{\text{tot\_copies}}$ varies from 10 to 32. Therefore problem grows rapidly and it is impossible to do a granular increase of the problem dimension.

Table 14 shows the optimal value (opt), the average one (avg), and the number of iterations for obtaining these (MC_step). Table 15 provides the p_{range}, the TTS and the considered p_{cons}. Also in this case, the choice of p_{cons} is done by considering the following policy: if the optimal solution is achieved by all solvers, it is fixed to 0.01%, while if even just one solver does not reach the expected energy, it is computed, exploiting also a saturation mechanism, for each optimization problem

TABLE 15: Results obtained by solving one hundred times nurse scheduling problems (`Nurse_`n`_`d) of different sizes (n-nurse and d-day, i.e. $(n \cdot d)$-variable) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 13. In this table, the probability of finding the final energy in a given range (p_{range}) and the TTS.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	$p_{\text{cons}}[\%]$	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**
Nurse_2_10	0.01	64.00	45.08	**85.00**	**24.27**	44.00	79.42	82.00	26.86	83.00	25.99
Nurse_2_15	0.01	**100.00**	**300.00**	**100.00**	**300.00**	**100.00**	**300.00**	**100.00**	**300.00**	**100.00**	**300.00**
Nurse_2_20	0.01	**100.00**	**400.00**	**100.00**	**400.00**	**100.00**	**400.00**	**100.00**	**400.00**	**100.00**	**400.00**
Nurse_3_10	0.01	**100.00**	**300.00**	**100.00**	**300.00**	**100.00**	**300.00**	**100.00**	**300.00**	**100.00**	**300.00**
Nurse_3_15	0.01	**100.00**	**450.00**	17.00	11121.83	11.00	17783.05	25.00	7203.53	33.00	5174.64
Nurse_3_20	0.01	**100.00**	**600.00**	47.00	4352.18	32.00	7164.56	53.00	3659.63	46.00	4484.20
Nurse_4_10	0.01	**100.00**	**400.00**	98.00	470.87	**100.00**	**400.00**	**100.00**	**400.00**	**100.00**	**400.00**
Nurse_4_15	0.01	71.00	2232.13	93.00	1039.05	98.00	706.31	96.00	858.41	**99.00**	**600.00**
Nurse_4_20	0.01	65.00	3509.30	94.00	1309.49	83.00	2079.13	96.00	1144.54	**98.00**	**941.75**
Nurse_5_10	6.67	1.00	229105.29	**8.00**	**27615.02**	2.00	113974.09	5.00	44890.57	7.00	31728.86
Nurse_10_10	20.00	0.00	460286721.69	**12.00**	**3602478.86**	0.00	46028672.17	0.00	46028672.17	0.00	4602867.22
Nurse_10_11	20.00	0.00	5063153.94	**1.00**	**504031.63**	0.00	5063153.94	0.00	5063153.94	0.00	5063153.94
Nurse_10_12	20.00	0.00	5523440.66	0.00	5523440.66	**1.00**	**549852.69**	0.00	5523440.66	**1.00**	**549852.69**
Nurse_10_13	20.00	0.00	5983727.38	3.00	196548.82	0.00	5983727.38	2.00	296332.62	**4.00**	**146654.31**
Nurse_10_14	20.00	0.00	6444014.10	**1.00**	**641494.81**	0.00	6444014.10	0.00	6444014.10	0.00	6444014.10
Nurse_10_15	20.00	0.00	6904300.83	**1.00**	**687315.86**	0.00	6904300.83	0.00	6904300.83	**1.00**	**687315.86**

TABLE 16: Setup configuration considered for solving one hundred times graph colouring problems (`GraphColouring_`y) of different sizes (three-colour, y-node) with the proposed algorithms and the SQA as a reference. The initial values for transverse fields and temperature are both equal to one for all the tests, while the number of involved copies differs according to the problem and is reported in the following.

Problem		$N_{\text{tot\_copies}}$	**SQA**	**SQPT**		**SQPA**		**SQPTPA1**			**SQPTPA2**		
Name	v_q		M	M	SYS	M	SYS	M	SYS_{temp}	SYS_{pop}	M	SYS_{temp}	SYS_{pop}
GraphColouring_3	9	10	10	2	5	2	5	2	3	2	2	3	2
GraphColouring_4	12	10	10	2	5	2	5	2	3	2	2	3	2
GraphColouring_5	15	10	10	2	5	2	5	2	3	2	2	3	2
GraphColouring_6	18	10	10	2	5	2	5	2	3	2	2	3	2
GraphColouring_7	21	10	10	2	5	2	5	2	3	2	2	3	2
GraphColouring_8	24	10	10	2	5	2	5	2	3	2	2	3	2
GraphColouring_9	27	10	10	2	5	2	5	2	3	2	2	3	2
GraphColouring_10	30	18	18	3	6	3	6	3	3	3	3	3	3
GraphColouring_31	93	32	32	4	8	4	8	4	4	4	4	4	4

TABLE 17: Results obtained by solving one hundred times graph colouring problems (`GraphColouring_`y) of different sizes (three-colour, y-node) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 16. In this table, the energy of the best solution between ones obtained by each solver in the one hundred repetitions is reported (**opt**) together with the average of the final energies found.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	**MC_step**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**
GraphColouring_3	9	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
GraphColouring_4	12	**0.00**	**0.00**	**0.00**	**0.00**	0.00	0.03	**0.00**	**0.00**	**0.00**	**0.00**
GraphColouring_5	15	0.00	0.01	**0.00**	**0.00**	0.00	0.10	**0.00**	**0.00**	0.00	0.02
GraphColouring_6	18	0.00	0.15	0.00	0.05	0.00	0.46	**0.00**	**0.04**	0.00	0.09
GraphColouring_7	21	0.00	1.04	0.00	0.43	0.00	1.51	0.00	0.51	**0.00**	**0.48**
GraphColouring_8	24	1.00	1.65	**1.00**	**1.49**	1.00	2.49	1.00	1.71	1.00	1.67
GraphColouring_9	27	0.00	1.21	0.00	0.86	0.00	1.89	**0.00**	**0.82**	0.00	0.93
GraphColouring_10	120	1.00	1.83	1.00	1.98	1.00	1.81	**1.00**	**1.72**	1.00	1.84
GraphColouring_31	37200	33.00	42.74	**16.00**	**18.37**	20.00	23.97	21.00	24.04	20.00	24.04

TABLE 18: Results obtained by solving one hundred times nurse scheduling problems graph colouring problems (`GraphColouring_`y) of different sizes (three-colour, y-node) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 16. In this table, the probability of finding the final energy in a given range (p_{range}) and the TTS.

Problem		SQA		SQPT		SQPA		SQPTPA1		SQPTPA2	
Name	$p_{\text{cons}}[\%]$	$p_{\text{range}}[\%]$	TTS	$p_{\text{range}}[\%]$	TTS	$p_{\text{range}}[\%]$	TTS	$p_{\text{range}}[\%]$	TTS	$p_{\text{range}}[\%]$	TTS
GraphColouring_3	0.10	**100.00**	**9.00**	**100.00**	**9.00**	**100.00**	**9.00**	**100.00**	**9.00**	**100.00**	**9.00**
GraphColouring_4	0.10	**100.00**	**12.00**	**100.00**	**12.00**	97.00	15.76	**100.00**	**12.00**	**100.00**	**12.00**
GraphColouring_5	0.10	99.00	15.00	**100.00**	**15.00**	90.00	30.00	**100.00**	**15.00**	98.00	17.66
GraphColouring_6	0.10	85.00	43.69	95.00	27.67	56.00	100.97	**96.00**	**25.75**	91.00	34.42
GraphColouring_7	0.10	21.00	410.26	61.00	102.71	15.00	595.06	56.00	117.80	**62.00**	**99.95**
GraphColouring_8	0.10	40.00	216.36	**55.00**	**138.41**	10.00	1049.01	38.00	231.20	42.00	202.90
GraphColouring_9	0.10	15.00	765.08	31.00	335.09	9.00	1318.40	**39.00**	**251.55**	29.00	363.05
GraphColouring_10	0.10	18.00	2784.67	12.00	4322.97	20.00	2476.52	**30.00**	**1549.37**	16.00	3169.54
GraphColouring_31	20.00	0.00	171226660.47	**97.00**	**48854.84**	0.00	171226660.47	0.00	171226660.47	0.00	171226660.47

according to the formula of equation 76. In this way, the value obtained (val) is sufficiently close to the actual optimal value to appreciate p_{range} lower than 100%. When the saturation mechanism is employed, some p_{range} are equal to 0%. This choice was done for guaranteeing a val which cannot cause too many p_{range} equal to 100%.
The $\frac{p_{\text{range}}}{\text{MC\_step}}$ and the TTS evolutions, varying the problem size (values obtained with problems of the same size were averaged) with a window w equal to 20, are reported in Figures 14e and 15e. The obtained TTS shape is as expected. In fact, it is possible to recognize, even if not in a particularly clean way due to the limited amount of samples, a square-root evolution with the problem size in the logarithmic axis.
Analyzing the obtained results, it is possible to say that the best solvers are the SQPT and the SQPTPA2, since they provide the highest p_{range} and the lowest TTS. At the same time, the SQA, and the SQPA solvers give the worst results, i.e. TTS is significantly higher than the others. Examples of cumulative distribution and energy evolutions for the 80-variable problem are reported in Figures 16e and 17e, respectively. It is possible to observe that the cumulative distributions of the SQPT, the SQPTPA1 and the SQPTPA2 are more concentrated on the left, i.e. the probability of reaching the optimal solution is higher. From the energy evolution point of view, it is possible to notice that all the algorithms have reached convergence. However, the SQA and the SQPA are significantly slower than the others.
Considering all the analyzed figures of merit, it is possible to conclude that the **SQPT and SQPTPA2 algorithms are the most suitable for exploring the nurse scheduling problems' objective function.**

6) Graph Colouring

Table 16 reports the setup configuration of the performed graph colouring tests. In particular, the number of Trotters, system copies and binary variables are shown for each problem solved. Each problem is identified with a name in the format: `GraphColouring_`y, where y is the number of nodes and the number of colours is fixed at three, and, consequently, the total number of involved variables is equal to $3 \cdot y$. In this case, the $N_{\text{tot\_copies}}$ grows from 10 to 32 with the problem size.
Table 17 shows the optimal value (opt), which is the lowest final value obtained by each solver in the one hundred runs considered, the average one (avg), which is obtained by averaging the one hundred obtained final values, and the number of iterations for obtaining these (MC_step).
Table 18 provides the p_{range}, the TTS and the considered p_{cons}. In case of achievement of the optimal solution by all solvers, the latter is fixed to 0.01%, while if even just one solver does not reach the expected energy, it is computed, exploiting also a saturation mechanism, for each optimization problem according to the formula reported in Equation 76. In this way, the value obtained (val) is sufficiently close to the actual optimal value to appreciate p_{range} lower than 100%. When the saturation mechanism is employed, some p_{range} are equal to 0%. This choice was done for guaranteeing a val which cannot cause too many p_{range} equal to 100%.
The $\frac{p_{\text{range}}}{\text{MC\_step}}$ and the TTS evolutions, varying the problem size (values obtained with problems of the same size were averaged) with a window w equal to 5, are reported in Figures 14f and 15f. The obtained TTS shape is coherent with expectations. In fact, it is possible to identify a square-root evolution with the problem size in the logarithmic axis.
Reading the obtained results, it is possible to say that the best solver is the SQPT, and the SQPTPA1, i.e. they provide the highest p_{range} and the lowest TTS. At the same time, the SQA, and the SQPA solvers give the worst results, i.e. TTS is significantly higher than the others. Examples of cumulative distribution and energy evolutions for the 30-variable problem are reported in Figures 16f and 17f respectively. It is possible to observe that the cumulative distributions of the SQPT, the SQPTPA1 and the SQPTPA2 are more concentrated on the left, i.e. the probability of reaching the optimal solution is higher. From the energy evolution point of view, it is possible to observe that all the algorithms except SQPT have reached convergence. However, the SQA and the SQPA are significantly slower than the others.

TABLE 19: Setup configuration considered for solving one hundred times minimum vertex cover problems (`MinimumVertexCover_`x) of different sizes (x-node) with the proposed algorithms and the SQA as a reference. The initial values for transverse fields and temperature are both equal to one for all the tests, while the number of involved copies differs according to the problem and is reported in the following.

Problem		$N_{\text{tot\_copies}}$	**SQA**	**SQPT**		**SQPA**		**SQPTPA1**			**SQPTPA2**		
Name	v_q		M	M	SYS	M	SYS	M	SYS_{temp}	SYS_{pop}	M	SYS_{temp}	SYS_{pop}
MinimumVertexCover_20	20	10	10	2	5	2	5	2	3	2	2	3	2
MinimumVertexCover_25	25	10	10	2	5	2	5	2	3	2	2	3	2
MinimumVertexCover_30	30	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_35	35	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_40	40	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_45	45	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_50	50	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_55	55	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_60	60	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_65	65	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_70	70	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_71	71	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_72	72	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_73	73	18	18	3	6	3	6	3	3	3	3	3	3
MinimumVertexCover_74	74	18	18	3	6	3	6	3	3	3	3	3	3

TABLE 20: Results obtained by solving one hundred times minimum vertex cover problems (`MinimumVertexCover_`x) of different sizes (x-node) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 19. In this table, the energy of the best solution between ones obtained by each solver in the one hundred repetitions is reported (**opt**) together with the average of the final energies found.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	**MC_step**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**
MinimumVertexCover_20	10	15.00	16.01	15.00	15.04	15.00	15.05	**15.00**	**15.00**	**15.00**	**15.00**
MinimumVertexCover_25	12	20.00	22.36	20.00	20.29	20.00	20.13	**20.00**	**20.07**	**20.00**	**20.06**
MinimumVertexCover_30	120	24.00	25.67	24.00	25.88	**24.00**	**25.34**	24.00	25.87	24.00	25.74
MinimumVertexCover_35	140	30.00	33.06	30.00	31.93	**28.00**	**31.64**	29.00	31.70	28.00	31.78
MinimumVertexCover_40	160	35.00	41.84	34.00	37.01	**34.00**	**36.77**	34.00	36.95	34.00	37.13
MinimumVertexCover_45	180	40.00	57.11	40.00	42.82	**39.00**	**43.13**	39.00	43.19	40.00	42.60
MinimumVertexCover_50	200	45.00	67.33	44.00	49.26	**42.00**	**48.23**	42.00	47.76	43.00	48.04
MinimumVertexCover_55	220	58.00	88.41	51.00	58.02	**48.00**	**53.53**	48.00	53.61	48.00	53.49
MinimumVertexCover_60	240	70.00	112.95	55.00	61.10	52.00	60.43	**52.00**	**59.01**	53.00	59.27
MinimumVertexCover_65	260	76.00	134.87	60.00	69.01	58.00	68.45	58.00	66.67	**57.00**	**65.46**
MinimumVertexCover_70	280	82.00	170.12	66.00	73.66	63.00	73.29	64.00	71.39	**64.00**	**70.80**
MinimumVertexCover_71	284	82.00	168.97	67.00	74.55	64.00	75.10	64.00	72.60	**64.00**	**72.52**
MinimumVertexCover_72	288	92.00	176.10	68.00	76.35	64.00	74.78	65.00	73.23	**64.00**	**71.86**
MinimumVertexCover_73	292	106.00	179.57	68.00	79.05	66.00	76.83	**66.00**	**74.16**	67.00	73.94
MinimumVertexCover_74	296	125.00	191.00	70.00	79.06	67.00	78.03	67.00	76.88	**67.00**	**75.32**

Considering all the analyzed figures of merit, it is possible to conclude that the **SQPTPA1 algorithm is the most suitable for exploring the graph colouring problems energy profile.**

7) Minimum Vertex Cover

Table 19 reports the setup configuration of the performed minimum vertex cover tests, in particular the number of Trotters, system copies and binary variables. Each problem is identified with a name in the format: `MinimumVertexCover_`x, where x is the number of nodes, i.e. the total number of involved variables. In this case, the $N_{\text{tot\_copies}}$ grows from 10 to 18 with the problem size. Table 20 shows the optimal value (opt), the average one (avg), and the number of iterations for obtaining these (MC_step). Table 21 provides the p_{range}, the TTS and the considered p_{cons}. In case of achievement of the optimal solution by all solvers, the latter is fixed to 0.01%, while if even just one solver does not reach the expected energy, it is computed, exploiting a saturation mechanism in both directions ($p_{\text{range}} \in$

TABLE 21: Results obtained by solving one hundred times minimum vertex cover problems (`MinimumVertexCover_`x) of different sizes (x-node) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 19. In this table, the probability of finding the final energy in a given range (p_{range}) and the TTS.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	$p_{\text{cons}}[\%]$	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**
MinimumVertexCover_20	0.10	25.00	160.08	97.00	13.13	96.00	14.31	**100.00**	**10.00**	**100.00**	**10.00**
MinimumVertexCover_25	0.10	3.00	1814.30	76.00	38.72	87.00	27.09	**94.00**	**19.64**	**94.00**	**19.64**
MinimumVertexCover_30	0.10	8.00	6627.61	4.00	13537.32	**14.00**	**3664.04**	9.00	5859.57	11.00	4742.15
MinimumVertexCover_35	6.67	5.00	12569.36	8.00	7732.21	**48.00**	**985.93**	31.00	1737.50	36.00	1444.64
MinimumVertexCover_40	2.86	6.00	11908.24	6.00	11908.24	**46.00**	**1195.79**	35.00	1710.44	28.00	2242.98
MinimumVertexCover_45	2.50	1.00	82477.90	2.00	41030.67	**31.00**	**2233.93**	22.00	3336.26	21.00	3516.56
MinimumVertexCover_50	6.67	1.00	91642.12	6.00	14885.30	**37.00**	**1993.43**	29.00	2689.23	26.00	3058.85
MinimumVertexCover_55	17.24	2.00	50148.60	60.00	1105.69	**94.00**	**360.11**	91.00	420.75	89.00	459.00
MinimumVertexCover_60	20.00	0.00	1104688.13	68.00	969.99	79.00	708.19	**82.00**	**644.53**	78.00	729.95
MinimumVertexCover_65	20.00	0.00	1196745.48	54.00	1541.92	63.00	1204.27	65.00	1140.52	**72.00**	**940.59**
MinimumVertexCover_70	20.00	0.00	1288802.82	74.00	957.22	68.00	1131.66	80.00	801.18	**83.00**	**727.70**
MinimumVertexCover_71	20.00	0.00	1307214.29	72.00	1027.42	56.00	1593.06	72.00	1027.42	**81.00**	**787.53**
MinimumVertexCover_72	20.00	0.00	1325625.76	59.00	1487.54	64.00	1298.18	74.00	984.57	**82.00**	**773.44**
MinimumVertexCover_73	20.00	0.00	1344037.23	74.00	998.25	63.00	1352.48	**87.00**	**659.10**	85.00	708.82
MinimumVertexCover_74	20.00	0.00	1362448.70	77.00	927.50	60.00	1487.66	73.00	1041.09	**81.00**	**820.80**

TABLE 22: Setup configuration considered for solving one hundred times number partitioning problems (`NumberPartitioning_`x) of different sizes (x-number) with the proposed algorithms and the SQA as a reference. The initial values for transverse fields and temperature are both equal to one for all the tests, while the number of involved copies differs according to the problem and is reported in the following.

Problem		$N_{\text{tot\_copies}}$	**SQA**	**SQPT**		**SQPA**		**SQPTPA1**			**SQPTPA2**		
Name	v_q		M	M	SYS	M	SYS	M	SYS_{temp}	SYS_{pop}	M	SYS_{temp}	SYS_{pop}
NumberPartitioning_10	10	10	10	2	5	2	5	2	3	2	2	3	2
NumberPartitioning_20	20	10	10	2	5	2	5	2	3	2	2	3	2
NumberPartitioning_30	30	10	10	2	5	2	5	2	3	2	2	3	2
NumberPartitioning_40	40	10	10	2	5	2	5	2	3	2	2	3	2
NumberPartitioning_50	50	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_60	60	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_70	70	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_80	80	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_90	90	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_100	100	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_110	110	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_120	120	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_130	130	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_140	140	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_150	150	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_160	160	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_170	170	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_180	180	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_190	190	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_200	200	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_210	210	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_220	220	18	18	3	6	3	6	3	3	3	3	3	3
NumberPartitioning_230	230	18	18	3	6	3	6	3	3	3	3	3	3

TABLE 23: Results obtained by solving one hundred times number partitioning problems (`NumberPartitioning_`x) of different sizes (x-number) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 22. In this table, the energy of the best solution between ones obtained by each solver in the one hundred repetitions is reported (**opt**) together with the average of the final energies found.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	**MC_step**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**	**opt**	**avg**
NumberPartitioning_10	10	**96100.00**	**939751.96**	96100.00	1080232.28	96100.00	1691727.16	96100.00	1375236.96	96100.00	1692447.16
NumberPartitioning_20	20	**1.00**	**500670.52**	9.00	719254.76	441.00	921029.24	81.00	710539.24	9.00	597222.12
NumberPartitioning_30	30	1.00	30789.88	25.00	36294.92	9.00	66638.68	**1.00**	**31594.92**	25.00	33995.64
NumberPartitioning_40	40	**4.00**	**93099.20**	16.00	82772.40	144.00	177001.24	16.00	153620.24	16.00	136018.36
NumberPartitioning_50	200	0.00	5548.00	0.00	5088.04	4.00	10990.56	**0.00**	**7557.48**	4.00	9020.40
NumberPartitioning_60	240	16.00	44570.80	**0.00**	**47638.20**	0.00	114826.52	4.00	86962.88	4.00	69656.64
NumberPartitioning_70	280	0.00	2865.40	**0.00**	**2883.20**	0.00	6820.12	4.00	3756.52	4.00	3123.96
NumberPartitioning_80	320	1.00	23157.64	**1.00**	**22475.72**	9.00	36160.12	1.00	31681.56	1.00	40771.72
NumberPartitioning_90	360	1.00	62042.04	1.00	62679.56	1.00	65237.08	**1.00**	**53520.84**	49.00	55889.72
NumberPartitioning_100	400	**0.00**	**34211.52**	4.00	32500.00	4.00	29220.32	4.00	33642.24	16.00	38132.00
NumberPartitioning_110	440	**1.00**	**5501.00**	1.00	7145.64	1.00	9657.48	1.00	8043.64	1.00	8803.64
NumberPartitioning_120	480	**0.00**	**8457.40**	**0.00**	**7120.96**	0.00	7555.12	0.00	9168.28	**0.00**	**10320.44**
NumberPartitioning_130	520	1.00	14621.08	25.00	16072.36	1.00	14013.32	1.00	14543.72	**1.00**	**13258.60**
NumberPartitioning_140	560	0.00	2432.44	**0.00**	**1797.44**	0.00	5318.48	0.00	3208.72	0.00	1569.68
NumberPartitioning_150	600	**1.00**	**1671.08**	1.00	1589.48	1.00	2693.72	**1.00**	**2052.44**	1.00	2048.76
NumberPartitioning_160	640	0.00	3476.16	**0.00**	**3131.20**	0.00	4669.08	**0.00**	**4245.92**	0.00	4034.52
NumberPartitioning_170	680	1.00	10426.76	1.00	9144.52	9.00	11383.40	**1.00**	**10645.88**	**1.00**	**14475.16**
NumberPartitioning_180	720	**0.00**	**1336.96**	0.00	1403.56	0.00	2962.32	0.00	1968.80	**0.00**	**2420.44**
NumberPartitioning_190	760	**0.00**	**1720.52**	**0.00**	**1571.64**	**0.00**	**2534.76**	0.00	3151.08	0.00	1861.00
NumberPartitioning_200	800	1.00	1651.80	**1.00**	**1964.92**	1.00	2352.76	1.00	2183.48	1.00	2260.52
NumberPartitioning_210	840	**0.00**	**4639.72**	4.00	4559.36	**0.00**	**6221.80**	4.00	6675.00	**0.00**	**5007.00**
NumberPartitioning_220	880	0.00	3057.40	0.00	4229.92	0.00	4257.44	**0.00**	**3259.92**	0.00	3616.68
NumberPartitioning_230	920	0.00	881.00	0.00	1343.40	0.00	3231.20	**0.00**	**1779.48**	0.00	4553.88

TABLE 24: Results obtained by solving one hundred times number partitioning problems (`NumberPartitioning_`x) of different sizes (x-number) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 22. In this table, the probability of finding the final energy in a given range (p_{range}) and the TTS.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	p_{cons}[%]	p_{range}[%]	**TTS**	p_{range}[%]	**TTS**	p_{range}[%]	**TTS**	p_{range}[%]	**TTS**	p_{range}[%]	**TTS**
NumberPartitioning_10	0.10	**71.00**	**37.20**	69.00	39.32	69.00	39.32	66.00	42.69	68.00	40.42
NumberPartitioning_20	20.00	**1.00**	**9164.21**	0.00	92057.34	0.00	92057.34	0.00	92057.34	0.00	92057.34
NumberPartitioning_30	20.00	2.00	6838.45	0.00	138086.02	0.00	138086.02	**4.00**	**3384.33**	0.00	138086.02
NumberPartitioning_40	20.00	**1.00**	**18328.42**	0.00	184114.69	0.00	184114.69	0.00	184114.69	0.00	184114.69
NumberPartitioning_50	20.00	1.00	91642.12	1.00	91642.12	0.00	920573.44	**3.00**	**30238.28**	0.00	920573.44
NumberPartitioning_60	20.00	0.00	1104688.13	**2.00**	**54707.56**	1.00	109970.54	0.00	1104688.13	0.00	1104688.13
NumberPartitioning_70	20.00	1.00	128298.96	**4.00**	**31587.08**	1.00	128298.96	0.00	1288802.82	0.00	1288802.82
NumberPartitioning_80	20.00	1.00	146627.38	**4.00**	**36099.52**	0.00	1472917.51	1.00	146627.38	2.00	72943.41
NumberPartitioning_90	20.00	2.00	82061.34	1.00	164955.81	4.00	40611.96	**5.00**	**32321.21**	0.00	1657032.20
NumberPartitioning_100	20.00	**1.00**	**183284.23**	0.00	1841146.89	0.00	1841146.89	0.00	1841146.89	0.00	1841146.89
NumberPartitioning_110	0.10	**6.00**	**32747.66**	1.00	201612.65	4.00	49636.84	4.00	49636.84	2.00	100297.20
NumberPartitioning_120	0.10	**2.00**	**109415.12**	**2.00**	**109415.12**	1.00	219941.08	1.00	219941.08	**2.00**	**109415.12**
NumberPartitioning_130	20.00	3.00	78619.53	0.00	2393490.95	1.00	238269.50	2.00	118533.05	**7.00**	**32998.01**
NumberPartitioning_140	0.10	2.00	127650.98	**5.00**	**50277.44**	3.00	84667.18	1.00	256597.92	1.00	256597.92
NumberPartitioning_150	0.10	**6.00**	**44655.90**	5.00	53868.68	4.00	67686.60	**6.00**	**44655.90**	5.00	53868.68
NumberPartitioning_160	0.10	3.00	96762.50	**4.00**	**72199.04**	1.00	293254.77	**4.00**	**72199.04**	2.00	145886.83
NumberPartitioning_170	20.00	1.00	311583.19	2.00	155004.76	0.00	3129949.71	**3.00**	**102810.15**	**3.00**	**102810.15**
NumberPartitioning_180	0.10	**5.00**	**64642.42**	1.00	329911.62	2.00	164122.68	4.00	81223.92	**5.00**	**64642.42**
NumberPartitioning_190	0.10	**3.00**	**114905.46**	**3.00**	**114905.46**	**3.00**	**114905.46**	2.00	173240.61	1.00	348240.04
NumberPartitioning_200	0.10	6.00	59541.21	**7.00**	**50766.17**	4.00	90248.80	4.00	90248.80	4.00	90248.80
NumberPartitioning_210	20.00	**1.00**	**384896.88**	0.00	3866408.46	**1.00**	**384896.88**	0.00	3866408.46	**1.00**	**384896.88**
NumberPartitioning_220	0.10	3.00	133048.43	2.00	200594.39	1.00	403225.31	**5.00**	**79007.40**	1.00	403225.31
NumberPartitioning_230	0.10	3.00	139096.09	5.00	82598.64	2.00	209712.32	**6.00**	**68472.39**	1.00	421553.73

TABLE 25: Setup configuration considered for solving one hundred times linear regression problems (`LinearRegression_`p) of different sizes (p-power, i.e. $4p$-variable) with the proposed algorithms and the SQA as a reference. The initial values for transverse fields and temperature are both equal to one for all the tests, while the number of involved copies differs according to the problem and is reported in the following.

Problem		$N_{\text{tot\_copies}}$	**SQA**	**SQPT**		**SQPA**		**SQPTPA1**			**SQPTPA2**		
Name	v_q		M	M	SYS	M	SYS	M	SYS_{temp}	SYS_{pop}	M	SYS_{temp}	SYS_{pop}
LinearRegression_5	20	10	10	2	5	2	5	2	3	2	2	3	2
LinearRegression_6	24	10	10	2	5	2	5	2	3	2	2	3	2
LinearRegression_7	28	10	10	2	5	2	5	2	3	2	2	3	2
LinearRegression_8	32	10	10	2	5	2	5	2	3	2	2	3	2
LinearRegression_9	36	10	10	2	5	2	5	2	3	2	2	3	2
LinearRegression_11	44	10	10	2	5	2	5	2	3	2	2	3	2
LinearRegression_12	48	10	10	2	5	2	5	2	3	2	2	3	2
LinearRegression_13	52	10	10	2	5	2	5	2	3	2	2	3	2
LinearRegression_14	56	10	10	2	5	2	5	2	3	2	2	3	2
LinearRegression_15	60	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_16	64	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_17	68	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_18	72	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_19	76	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_20	80	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_25	100	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_35	140	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_45	180	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_60	240	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_70	280	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_80	320	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_90	360	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_100	400	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_110	440	18	18	3	6	3	6	3	3	3	3	3	3
LinearRegression_120	480	18	18	3	6	3	6	3	3	3	3	3	3

$[0.01\%, 20\%]$), for each optimization problem according to the following formula:

$$p_{\text{cons}}[\%] = \left|1 - \frac{\min_h}{\min_l}\right| 100\,, \tag{77}$$

where $\min_h$ is the highest opt among ones of all the analyzed solvers, while $\min_l$ is the lowest one. In this way, the value obtained (val) is sufficiently close to the actual optimal value to appreciate p_{range} lower than 100%. When the saturation mechanism is employed, some p_{range} are equal to 0%. This choice was done for guaranteeing a val which cannot cause too many p_{range} equal to 100%.

The $\frac{p_{\text{range}}}{\text{MC\_step}}$ and the TTS evolutions, varying the problem size with a window w equal to 50, are reported in Figures 14g and 15g. The obtained TTS shape is coherent with expectations. In fact, it is possible to identify a square-root evolution with the problem size in the logarithmic axis.

Reading the obtained results, it is possible to say that the best solver for middle-size problems is the SQPA, while for small and large ones the best are the SQPTPA1 and the SQPTPA2, i.e. they provide the highest p_{range} and the lowest TTS. At the same time, the SQA, and the SQPT solvers provide the worst results for all problem sizes, i.e. TTS is significantly higher than the others. Examples of cumulative distribution and energy evolutions for the 31-variable problem are reported in Figures 16g and 17g, respectively. It is possible to observe that the cumulative distribution of the SQPA is more concentrated on the left, i.e. the probability of reaching the optimal solution is higher. From the energy evolution point of view, it is possible to observe that all the algorithms have reached convergence.

Considering all the analyzed figures of merit, it is possible to conclude that the **SQPTPA2 algorithm is the most suitable for exploring the minimum vertex cover problems energy profile.**

8) Number Partitioning

Table 22 reports the algorithms configurations chosen for solving number partitioning tests. In particular, the number of Trotters, system copies and binary variables are shown. Each problem is identified with a name in the format: `NumberPartitioning_`x, where x is the number of numbers, i.e. the total number of involved variables. In this case, the $N_{\text{tot\_copies}}$ grows from 10 to 18 with the problem size.

Table 23 shows the optimal value (opt), the average one (avg), and the number of iterations for obtaining these (MC_step).

Table 24 provides the p_{range}, the TTS and the considered p_{cons}. In case of achievement of the optimal solution by all solvers, the latter is fixed to 0.01%, while if even just one

TABLE 26: Results obtained by solving one hundred times linear regression problems (`LinearRegression_`p) of different sizes (p-power, i.e. $4p$-variable) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 25. In this table, the energy of the best solution between ones obtained by each solver in the one hundred repetitions is reported (**opt**) together with the average of the final energies found.

Problem		SQA		SQPT		SQPA		SQPTPA1		SQPTPA2	
Name	MC_step	opt	avg	opt	avg	opt	avg	opt	avg	opt	avg
LinearRegression_5	10	-3565.51	-3565.41	**-3565.51**	**-3565.26**	-3565.51	-3563.97	-3565.51	-3565.44	-3565.51	-3565.42
LinearRegression_6	12	-35.28	-35.28	**-35.28**	**-35.28**	-35.28	-35.25	-35.28	-35.28	-35.28	-35.28
LinearRegression_7	14	-2194.86	-2194.85	-2194.86	-2194.86	-2194.86	-2194.82	-2194.86	-2194.86	**-2194.86**	**-2194.86**
LinearRegression_8	16	-3449.68	-3449.67	**-3449.68**	**-3449.66**	-3449.68	-3449.31	-3449.68	-3449.68	-3449.68	-3449.68
LinearRegression_9	18	-574.65	-574.63	-574.65	-574.64	-574.65	-574.63	-574.65	-574.64	**-574.65**	**-574.64**
LinearRegression_11	22	-2054.22	-2054.20	-2054.22	-2054.21	-2054.22	-2054.20	-2054.22	-2054.21	**-2054.22**	**-2054.21**
LinearRegression_12	24	-158.52	-158.50	-158.52	-158.52	-158.52	-158.49	-158.52	-158.52	**-158.52**	**-158.52**
LinearRegression_13	26	-43.79	-43.77	-43.79	-43.78	-43.79	-43.77	-43.79	-43.78	**-43.79**	**-43.78**
LinearRegression_14	28	-1268.22	-1268.20	-1268.22	-1268.20	-1268.22	-1268.17	-1268.22	-1268.21	**-1268.22**	**-1268.20**
LinearRegression_15	300	-174.63	-174.63	-174.63	-174.63	**-174.63**	**-174.63**	-174.63	-174.63	-174.63	-174.63
LinearRegression_16	32	-3345.13	-3345.13	-3345.13	-3345.12	**-3345.13**	**-3345.12**	-3345.13	-3345.12	-3345.13	-3345.12
LinearRegression_17	34	**-181.54**	**-181.54**	-181.54	-181.53	-181.54	-181.53	-181.54	-181.53	-181.54	-181.53
LinearRegression_18	36	-129.89	-129.89	-129.89	-129.88	-129.89	-129.89	**-129.89**	**-129.89**	-129.89	-129.89
LinearRegression_19	38	-2234.23	-2234.22	-2234.23	-2234.22	**-2234.23**	**-2234.23**	-2234.23	-2234.22	-2234.23	-2234.23
LinearRegression_20	40	-2405.43	-2405.43	-2405.43	-2405.42	-2405.43	-2405.43	-2405.43	-2405.43	**-2405.43**	**-2405.43**
LinearRegression_25	500	-159.27	-159.27	-159.27	-159.27	**-159.27**	**-159.27**	-159.27	-159.27	-159.27	-159.27
LinearRegression_35	700	-189.11	-189.10	-189.11	-189.10	**-189.11**	**-189.11**	-189.11	-189.11	-189.11	-189.11
LinearRegression_45	900	-1379.65	-1379.65	-1379.65	-1379.65	**-1379.65**	**-1379.65**	-1379.65	-1379.65	-1379.65	-1379.65
LinearRegression_60	120	-1310.37	-1310.36	-1310.37	-1310.36	**-1310.37**	**-1310.37**	-1310.37	-1310.36	-1310.37	-1310.36
LinearRegression_70	140	-3518.65	-3518.65	-3518.65	-3518.65	**-3518.65**	**-3518.65**	-3518.65	-3518.65	-3518.65	-3518.65
LinearRegression_80	160	-1284.62	-1284.62	-1284.62	-1284.62	**-1284.62**	**-1284.62**	-1284.62	-1284.62	-1284.62	-1284.62
LinearRegression_90	180	-2287.15	-2287.14	-2287.15	-2287.14	**-2287.15**	**-2287.14**	-2287.15	-2287.14	-2287.15	-2287.14
LinearRegression_100	200	-156.18	-156.18	-156.18	-156.18	**-156.18**	**-156.18**	-156.18	-156.18	-156.18	-156.18
LinearRegression_110	220	-235.21	-235.21	-235.21	-235.21	**-235.21**	**-235.21**	-235.21	-235.21	-235.21	-235.21
LinearRegression_120	240	-1246.65	-1246.65	-1246.65	-1246.65	**-1246.65**	**-1246.65**	-1246.65	-1246.65	-1246.65	-1246.65

solver does not reach the expected energy, it is computed, exploiting a saturation mechanism, for each optimization problem according to the formula reported in Equation 76. In this way, the value obtained (val) is sufficiently close to the actual optimal value to appreciate p_{range} lower than 100%. When the saturation mechanism is employed, some p_{range} are equal to 0%. This choice was done for guaranteeing a val which cannot cause too many p_{range} equal to 100%.
The $\frac{p_{\text{range}}}{\text{MC\_step}}$ and the TTS evolutions, varying the problem size with a window w equal to 50, are reported in Figures 14h and 15h. The obtained TTS shape is coherent with expectations. In fact, it is possible to identify a square-root evolution with the problem size in the logarithmic axis.
Reading the obtained results, it is possible to notice that the SQA and the SQPT are the best solvers, while the SQPA and the SQPTPA2 solvers give the worst results, i.e. TTS is significantly higher than the others. Examples of cumulative distribution and energy evolutions for the 170-variable problem are reported in Figures 16h and 17h respectively. It is possible to notice that the cumulative distribution of the SQPT is more concentrated on the left, i.e. the probability of reaching the optimal solution is higher. From the energy evolution point of view, it is possible to observe that the SQPTPA2 has not reached convergence and the SQPA reaches a lower energy point at the beginning, starting, however, to grow for converging in a higher energy configuration.
Considering all the analyzed figures of merit, it is possible to conclude that the **SQA and the SQPT algorithms are the most suitable for exploring the number partitioning problems energy profile.**

9) Linear regression

Table 25 reports the setup configuration of the performed linear regression tests. In particular, it shows the number of Trotters, system copies and binary variables. Each problem is identified with a name in the format: `LinearRegression_`p, where p is the number of powers, i.e. the total number of involved variables is equal to $4p$. In this case, the $N_{\text{tot\_copies}}$ grows from 10 to 18 with the problem size.
Table 26 shows the optimal value (opt), the average one (avg), and the number of iterations for obtaining these (MC_step).
Table 27 provides the p_{range}, the TTS and the considered p_{cons}. In case of achievement of the optimal solution by all, it is

TABLE 27: Results obtained by solving one hundred times linear regression problems (`LinearRegression_`p) of different sizes (p-power, i.e. $4p$-variable) with the proposed algorithms and the SQA as a reference. Test setup configurations are reported in Table 25. In this table, the energy of the best solution between ones obtained by each solver in the one hundred repetitions is reported (**opt**) together with the average of the final energies found.

Problem		**SQA**		**SQPT**		**SQPA**		**SQPTPA1**		**SQPTPA2**	
Name	$p_{\text{cons}}[\%]$	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**	$p_{\text{range}}[\%]$	**TTS**
LinearRegression_5	1.00e-05	86.00	23.42	**94.00**	**16.37**	57.00	54.57	87.00	22.57	82.00	26.86
LinearRegression_6	1.00e-05	99.00	12.00	**100.00**	**12.00**	93.00	20.78	98.00	14.13	96.00	17.17
LinearRegression_7	1.00e-05	79.00	41.31	99.00	14.00	91.00	26.77	98.00	16.48	**100.00**	**14.00**
LinearRegression_8	1.00e-05	59.00	82.64	**91.00**	**30.60**	52.00	100.39	90.00	32.00	90.00	32.00
LinearRegression_9	1.00e-05	13.00	595.23	28.00	252.33	24.00	302.05	40.00	162.27	**49.00**	**123.11**
LinearRegression_11	1.00e-05	4.00	2481.84	41.00	192.02	27.00	321.93	41.00	192.02	**50.00**	**146.16**
LinearRegression_12	2.11e-05	1.00	10997.05	9.00	1171.91	16.00	633.91	17.00	593.16	**20.00**	**495.30**
LinearRegression_13	1.00e-05	1.00	11913.47	4.00	2933.09	9.00	1269.57	10.00	1136.43	**11.00**	**1027.47**
LinearRegression_14	1.00e-05	1.00	12829.90	2.00	6382.55	11.00	1106.50	12.00	1008.69	**19.00**	**611.92**
LinearRegression_15	1.00e-05	5.00	26934.34	5.00	26934.34	**13.00**	**9920.51**	7.00	19037.31	1.00	137463.17
LinearRegression_16	1.00e-05	7.00	2030.65	2.00	7294.34	**8.00**	**1767.36**	4.00	3609.95	6.00	2381.65
LinearRegression_17	4.10e-05	**3.00**	**5140.51**	1.00	15579.16	2.00	7750.24	2.00	7750.24	2.00	7750.24
LinearRegression_18	4.91e-05	1.00	16495.58	1.00	16495.58	1.00	16495.58	**2.00**	**8206.13**	1.00	16495.58
LinearRegression_19	1.00e-05	5.00	3411.68	2.00	8662.03	**9.00**	**1855.53**	4.00	4286.82	8.00	2098.74
LinearRegression_20	1.00e-05	9.00	1953.19	5.00	3591.25	10.00	1748.35	12.00	1440.99	**15.00**	**1133.45**
LinearRegression_25	1.00e-05	5.00	44890.57	1.00	229105.29	**12.00**	**18012.39**	3.00	75595.70	8.00	27615.02
LinearRegression_35	2.75e-05	1.00	320747.40	1.00	320747.40	**32.00**	**8358.65**	18.00	16243.91	18.00	16243.91
LinearRegression_45	1.00e-05	28.00	12616.75	10.00	39337.82	**58.00**	**4777.70**	47.00	6528.26	31.00	11169.65
LinearRegression_60	1.00e-05	2.00	27353.78	1.00	54985.27	**12.00**	**4322.97**	9.00	5859.57	10.00	5245.04
LinearRegression_70	1.00e-05	14.00	4274.71	7.00	8884.08	**18.00**	**3248.78**	9.00	6836.17	10.00	6119.22
LinearRegression_80	1.00e-05	9.00	7812.77	1.00	73313.69	**16.00**	**4226.06**	14.00	4885.38	13.00	5290.94
LinearRegression_90	1.00e-05	28.00	2523.35	8.00	9941.41	**47.00**	**1305.65**	37.00	1794.08	30.00	2324.05
LinearRegression_100	5.08e-05	7.00	12691.54	1.00	91642.12	**17.00**	**4943.04**	10.00	8741.74	15.00	5667.24
LinearRegression_110	1.00e-05	2.00	50148.60	1.00	100806.33	**3.00**	**33262.11**	2.00	50148.60	1.00	100806.33
LinearRegression_120	1.00e-05	12.00	8645.95	6.00	17862.36	**27.00**	**3511.93**	11.00	9484.29	17.00	5931.64

fixed to 0.1% for each optimization problem, while in the others it is computed according to the formula of Equation 76. In this way, the value obtained (val) is sufficiently close to the actual optimal value to appreciate p_{range} lower than 100%. The $\frac{p_{\text{range}}}{\text{MC\_step}}$ and the TTS evolutions, varying the problem size with a window w equal to 10, are reported in Figures 14i and 15i. The obtained TTS shape is consistent with expectations. In fact, it is possible to identify a square-root evolution with the problem size in the logarithmic axis.

Reading the obtained results, it is possible to notice that, for large-size problems, the SQPA is the best solver, while the SQA and the SQPT solvers give good results for small-size problems, but with larger ones provide the worst, i.e. TTS is significantly higher than the others. The SQPTPA1 and the SQPTAPA2 have good performance for all problem sizes.

Examples of cumulative distribution and energy evolutions for the 400-variable problem are reported in Figures 16i and 17i, respectively. It is possible to notice that the cumulative distribution of the SQPA and the SQPTPA1 are more concentrated on the left, i.e. the probability of reaching the optimal solution is higher. From the energy evolution point of view, it is possible to observe that all the solvers have reached convergence. However, the SQA and the SQPT reached a higher convergence value than the other. Moreover, the SQPTPA1 and the SQPTPA2 have a faster evolution than SQPA.

Considering all the analyzed figures of merit, it is possible to conclude that the **SQA, SQPTPA1 and the SQPTPA2 algorithms are the most suitable for exploring the linear regression problems energy profile.**

V. CONCLUSION

This work proposed four new hybrid quantum-classical algorithms obtained from the combination of SQA, PT and PA and compared them with SQA solving many types of QUBO problems. The results show that the proposed approaches can significantly improve the quality of the results with equal iterations with respect to SQA. However, the best solver is strongly correlated with the energy profile characteristics of the target optimization problem, coherently with expectation. For example, the SQPTPA1 is the best solver for max-cut problems, while the SQPT provides the best results with graph colouring optimization. Nevertheless, on average, the SQPTPA1 and the SQPTPA2 give a significantly good performance thanks to their capability to exploit the advantages of both SQA, PT and PA solution space exploration. Therefore, these approaches can be defined as promising.

The software implementations of the approaches considered

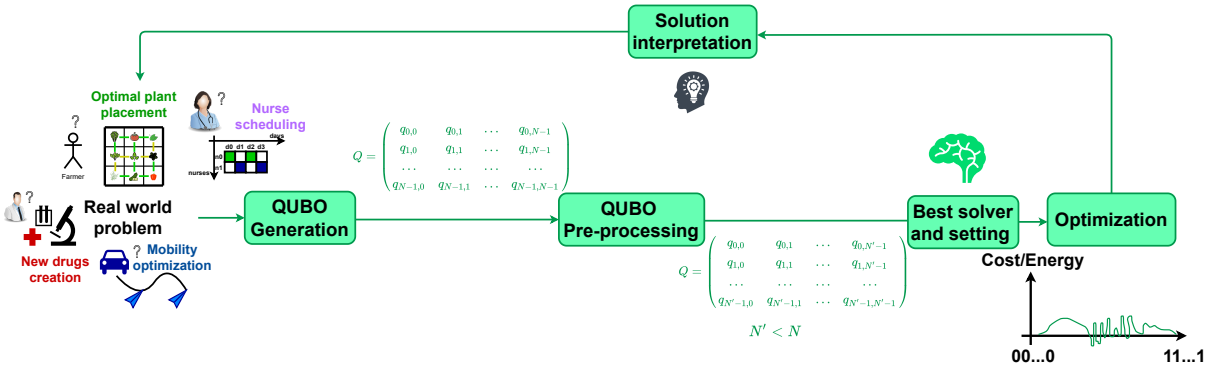


FIGURE 18: Automatic toolchain for QUBO solving. Staring from a real-world problem in an abstract description, the *QUBO generation* block provides the QUBO formulation. Then, the *QUBO pre-processing* block tries to reduce the problem dimension by fixing some variables. The *Best solver and setting* block identifies the best solver for the problem energy profile based on previous experience, establishes the value of algorithm degrees of freedom and, eventually, performs an additional solver-aware pre-processing. Furthermore, the solver looks for the optimal solution (*Optimization* step). Finally, the solution found is interpreted and applied to the starting real-world application (*Solution interpretation*).

in this work have to be seen as a proof-of-concept of the exploration mechanisms' validity. In order to exploit the potential of the new proposed algorithm, it will be necessary to obtain optimized and parallelized hardware implementations as possible. This should also allow the extension of the benchmark analysis to larger optimizations. Moreover, they can become more effective if a pre-conditioning procedure is applied to the QUBO problem for obtaining an energy profile more compatible with the target solver exploration.

Even though the current status of the work is preliminary, the performed analysis to evaluate the solver's effectiveness in optimizing many types of problems is a fundamental milestone for obtaining an automatic toolchain which can help the QUBO-solving procedure. Indeed, from a long-term perspective, the main idea is to develop a structure like the one reported in Figure 18, i.e. capable of managing any step of quantum-compliant optimization. In particular, the first step will manage the conversion from a real-world problem to a QUBO formulation, as in [101]. The second will be a solver-unaware pre-processing step for reducing the number of involved binary variables. The third will be the solver selection — among quantum, quantum-inspired and hybrid quantum-classical ones — based on problem energy profile characteristics, previous experience and further solver-aware pre-processing. The solver selection could also include a preliminary step associated with the degrees of freedom management, like that reported for the quantum circuit model in [35], to further improve the exploration quality. Finally, the solution space will be explored, detecting the optimal solution and interpreting it.

Even though the project seems particularly ambitious, we will strive to complete the toolchain to help researchers and industries in solving optimization problems which can improve people's lives. We hope that developing the backend prototypes discussed in this article can be a good starting point for achieving the goal.

REFERENCES

[1] H. Sakaguchi, K. Ogata, T. Isomura, S. Utsunomiya, Y. Yamamoto, and K. Aihara, "Boltzmann sampling by degenerate optical parametric oscillator network for structure-based virtual screening," Entropy, vol. 18, no. 10, p. 365, 2016. https://doi.org/10.3390/e18100365.

[2] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, "An application of combinatorial optimization to statistical physics and circuit layout design," Operations Research, vol. 36, no. 3, pp. 493–513, 1988. https://doi.org/10.1287/opre.36.3.493.

[3] H. Zheng, Y. Feng, and J. Tan, "A hybrid energy-aware resource allocation approach in cloud manufacturing environment," IEEE Access, vol. 5, pp. 12648–12656, 2017. https://doi.org/10.1109/ACCESS.2017.2715829.

[4] S. Kumar, P. Jangir, G. G. Tejani, M. Premkumar, and H. H. Alhelou, "Mopgo: A new physics-based multi-objective plasma generation optimizer for solving structural optimization problems," IEEE Access, vol. 9, pp. 84982–85016, 2021. https://doi.org/10.1109/ACCESS.2021.3087739.

[5] K. Wils, "Quantum computing for structural optimization," 2020. http://resolver.tudelft.nl/uuid:5167107a-7c26-4779-825a-e99702949870.

[6] A. D. Boursianis, M. S. Papadopoulou, M. Salucci, A. Polo, P. Sarigiannidis, K. Psannis, S. Mirjalili, S. Koulouridis, and S. K. Goudos, "Emerging swarm intelligence algorithms and their applications in antenna design: The gwo, woa, and ssa optimizers," Applied Sciences, vol. 11, no. 18, p. 8330, 2021. https://www.mdpi.com/2076-3417/11/18/8330.

[7] J. Fliege, L. G. Drummond, and B. F. Svaiter, "Newton's method for multiobjective optimization," SIAM Journal on Optimization, vol. 20, no. 2, pp. 602–626, 2009. https://doi.org/10.1137/08071692X.

[8] Ž. Povalej, "Quasi-newton's method for multiobjective optimization," Journal of Computational and Applied Mathematics, vol. 255, pp. 765–777, 2014. https://doi.org/10.1016/j.cam.2013.06.045.

[9] C. Audet and J. E. Dennis Jr, "Analysis of generalized pattern searches," SIAM Journal on optimization, vol. 13, no. 3, pp. 889–903, 2002. https://doi.org/10.1137/S1052623400378742.

[10] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," Soft computing, vol. 22, no. 2, pp. 387–408, 2018. https://doi.org/10.1007/s00500-016-2474-6.

[11] C. Blum, "Ant colony optimization: Introduction and recent trends," Physics of Life reviews, vol. 2, no. 4, pp. 353–373, 2005. https://doi.org/10.1016/j.plrev.2005.10.001.

[12] R. Rao and V. Patel, "A multi-objective improved teaching-learning based optimization algorithm for unconstrained and constrained optimization problems," International Journal of Industrial Engineering Computations, vol. 5, no. 1, pp. 1–22, 2014. https://doi.org/10.5267/j.ijiec.2013.09.007.

[13] S. Kumar, G. G. Tejani, N. Pholdee, S. Bureerat, and P. Jangir, "Multi-objective teaching-learning-based optimization for structure optimiza-

tion," Smart Science, vol. 10, no. 1, pp. 56–67, 2022. https://doi.org/10.1080/23080477.2021.1975074.

[14] Z. Li, X. Lin, Q. Zhang, and H. Liu, "Evolution strategies for continuous optimization: A survey of the state-of-the-art," Swarm and Evolutionary Computation, vol. 56, p. 100694, 2020. https://doi.org/10.1016/j.swevo.2020.100694.

[15] J. F. Frenzel, "Genetic algorithms," IEEE potentials, vol. 12, no. 3, pp. 21–24, 1993. https://doi.org/10.1109/45.282292.

[16] P. Singh and P. Dwivedi, "Integration of new evolutionary approach with artificial neural network for solving short-term load forecast problem," Applied Energy, vol. 217, pp. 537–549, 2018. https://doi.org/10.1016/j.apenergy.2018.02.131.

[17] P. Singh, R. Kottath, and G. G. Tejani, "Ameliorated follow the leader: Algorithm and application to truss design problem," Structures, vol. 42, pp. 181–204, 2022. https://doi.org/10.1016/j.istruc.2022.05.105.

[18] X. Wang, X. Z. Gao, and S. J. Ovaska, "Artificial immune optimization methods and applications-a survey," in 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583), vol. 4, pp. 3415–3420, IEEE, 2004. https://doi.org/10.1109/ICSMC.2004.1400870.

[19] L. Li, Q. Lin, and Z. Ming, "A survey of artificial immune algorithms for multi-objective optimization," Neurocomputing, vol. 489, pp. 211–229, 2022. https://doi.org/10.1016/j.neucom.2021.08.154.

[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, pp. 671–680, May 1983. https://doi.org/10.1126/science.220.4598.671.

[21] S. Niu, A. Suau, G. Staffelbach, and A. Todri-Sanial, "A hardware-aware heuristic for the qubit mapping problem in the NISQ era," IEEE Transactions on Quantum Engineering, vol. 1, pp. 1–14, 2020. https://doi.org/10.1109/TQE.2020.3026544.

[22] P. Ghannadi, S. Kourehli, and S. Mirjalili, "A review of the application of the simulated annealing algorithm in structural health monitoring (1995-2021)," Frattura ed Integrità Strutturale, vol. 64, pp. 51–76, 2023. https://doi.org/10.3221/IGF-ESIS.64.04.

[23] F. He and Q. Ye, "A bearing fault diagnosis method based on wavelet packet transform and convolutional neural network optimized by simulated annealing algorithm," Sensors, vol. 22, no. 4, 2022. https://doi.org/10.3390/s22041410.

[24] Y. Liu, A. A. Heidari, Z. Cai, G. Liang, H. Chen, Z. Pan, A. Alsufyani, and S. Bourouis, "Simulated annealing-based dynamic step shuffled frog leaping algorithm: Optimal performance design and feature selection," Neurocomputing, vol. 503, pp. 325–362, 2022. https://doi.org/10.1016/j.neucom.2022.06.075.

[25] "Lithium-ion batteries health prognosis via differential thermal capacity with simulated annealing and support vector regression," Energy, vol. 250, p. 123829, 2022. https://doi.org/10.1016/j.energy.2022.123829.

[26] S. Kumar, G. G. Tejani, N. Pholdee, and S. Bureerat, "Performance enhancement of meta-heuristics through random mutation and simulated annealing-based selection for concurrent topology and sizing optimization of truss structures," Soft Computing, vol. 26, no. 12, pp. 5661–5683, 2022. https://doi.org/10.1007/s00500-022-06930-2.

[27] S. Kumar, G. G. Tejani, N. Pholdee, and S. Bureerat, "Improved meta-heuristics through migration-based search and an acceptance probability for truss optimization," Asian Journal of Civil Engineering, vol. 21, no. 7, pp. 1217–1237, 2020. https://doi.org/10.1007/s42107-020-00271-x.

[28] D. B. Fontes, S. M. Homayouni, and J. F. Gonçalves, "A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources," European Journal of Operational Research, vol. 306, no. 3, pp. 1140–1157, 2023. https://doi.org/10.1016/j.ejor.2022.09.006.

[29] S. Kumar, P. Jangir, G. G. Tejani, and M. Premkumar, "Moteo: A novel physics-based multiobjective thermal exchange optimization algorithm to design truss structures," Knowledge-Based Systems, vol. 242, p. 108422, 2022. https://doi.org/10.1016/j.knosys.2022.108422.

[30] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," Physical Review E, vol. 58, no. 5, p. 5355, 1998. https://doi.org/10.1103/PhysRevE.58.5355.

[31] B. K. Chakrabarti, H. Leschke, P. Ray, T. Shirai, and S. Tanaka, "Quantum annealing and computation: challenges and perspectives," Philosophical Transactions of the Royal Society A, vol. 381, no. 2241, p. 20210419, 2023. https://doi.org/10.1098/rsta.2021.0419.

[32] A. Rajak, S. Suzuki, A. Dutta, and B. K. Chakrabarti, "Quantum annealing: an overview," Philosophical Transactions of the Royal Society A, vol. 381, no. 2241, p. 20210417, 2023. https://doi.org/10.1098/rsta.2021.0417.

[33] M. A. Nielsen and I. L. Chuang, Quantum computation and quantum information. Cambridge ; New York: Cambridge University Press, 10th anniversary ed ed., 2010.

[34] C. Durr and P. Hoyer, "A quantum algorithm for finding the minimum," 1996. https://doi.org/10.48550/ARXIV.QUANT-PH/9607014.

[35] L. Giuffrida, D. Volpe, G. A. Cirillo, M. Zamboni, and G. Turvani, "Engineering grover adaptive search: Exploring the degrees of freedom for efficient QUBO solving," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 12, no. 3, pp. 614–623, 2022. https://doi.org/10.1109/JETCAS.2022.3202566.

[36] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," Nature Communications, jul 2014. https://doi.org/10.1038/ncomms5213.

[37] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014. https://doi.org/10.48550/ARXIV.1411.4028.

[38] T. Liu, J. Sun, G. Wang, and Y. Lu, "A multi-objective quantum genetic algorithm for mimo radar waveform design," Remote Sensing, vol. 14, no. 10, 2022. https://doi.org/10.3390/rs14102387.

[39] J. Preskill, "Quantum Computing in the NISQ era and beyond," Quantum, vol. 2, p. 79, Aug. 2018.

[40] H. Goto, K. Tatsumura, and A. R. Dixon, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems," Science advances, vol. 5, no. 4, p. eaav2372, 2019. https://doi.org/10.1126/sciadv.aav2372.

[41] S. Matsubara, M. Takatsu, T. Miyazawa, T. Shibasaki, Y. Watanabe, K. Takemoto, and H. Tamura, "Digital annealer for high-speed solving of combinatorial optimization problems and its applications," in 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 667–672, 2020. https://doi.org/10.1109/ASP-DAC47756.2020.9045100.

[42] E. Crosson and A. W. Harrow, "Simulated quantum annealing can be exponentially faster than classical simulated annealing," in 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), pp. 714–723, 2016. https://doi.org/10.1109/FOCS.2016.81.

[43] N. Chancellor, "Modernizing quantum annealing using local searches," New Journal of Physics, vol. 19, no. 2, p. 023024, 2017. https://doi.org/10.1088/1367-2630/aa59c4.

[44] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using QUBO models," 2018. https://doi.org/10.48550/arXiv.1811.11538.

[45] B. Dury and O. Di Matteo, "A QUBO formulation for qubit allocation," 2020. https://doi.org/10.48550/ARXIV.2009.00140.

[46] S. Speziali, F. Bianchi, A. Marini, L. Menculini, M. Proietti, L. F. Termite, A. Garinei, M. Marconi, and A. Delogu, "Solving sensor placement problems in real water distribution networks using adiabatic quantum computation," 2021. https://doi.org/10.48550/arxiv.2108.04075.

[47] T. D. Tambunan, A. B. Suksmono, I. J. M. Edward, and R. Mulyawan, "Quantum annealing for vehicle routing problem with weighted segment," 2022. https://doi.org/10.48550/ARXIV.2203.13469.

[48] K. Ikeda, Y. Nakamura, and T. S. Humble, "Application of quantum annealing to nurse scheduling problem," Scientific reports, vol. 9, no. 1, pp. 1–10, 2019. https://doi.org/10.48550/arXiv.1904.12139.

[49] M. Ayodele, R. Allmendinger, M. López-Ibáñez, and M. Parizy, "Multi-objective qubo solver: Bi-objective quadratic assignment problem," in Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '22, (New York, NY, USA), p. 467–475, Association for Computing Machinery, 2022. https://doi.org/10.1145/3512290.3528698.

[50] Y. NAITO and K. FUJIYOSHI, "A study on updating spins in ising model to solve combinatorial optimization problems," city, vol. 4, p. 3. https://sasimi.jp/new/sasimi2019/files/archive/pdf/p310_R4-13.pdf.

[51] K. Tanahashi, S. Takayanagi, T. Motohashi, and S. Tanaka, "Application of ising machines and a software development for ising machines," Journal of the Physical Society of Japan, vol. 88, no. 6, p. 061010, 2019. https://doi.org/10.7566/JPSJ.88.061010.

[52] J. T. Iosue, "qubovert documentation," 2019. [Online at https://qubovert.readthedocs.io/en/stable/,; accessed 10-May-2022].

[53] J. T. Iosue, "pyqubo." https://github.com/recruit-communication.

[54] M. Zaman, K. Tanahashi, and S. Tanaka, "PyQUBO: Python library for mapping combinatorial optimization problems to QUBO form," IEEE Transactions on Computers, vol. 71, no. 4, pp. 838–850, 2022. https://doi.org/10.1109/TC.2021.3063618.

[55] "Python package dimod." https://github.com/dwavesystems/dimod.

[56] C. Cook, H. Zhao, T. Sato, M. Hiromoto, and S. X.-D. Tan, "GPU-based Ising computing for solving max-cut combinatorial optimization problems," Integration, vol. 69, pp. 335–344, 2019. https://doi.org/10.1016/j.vlsi.2019.07.003.

[57] Y. Haribara, S. Utsunomiya, K.-i. Kawarabayashi, and Y. Yamamoto, "A coherent ising machine for max-cut problems: Performance evaluation against semidefinite programming relaxation and simulated annealing," arXiv preprint arXiv:1501.07030, 2015. https://doi.org/10.1007/978-4-431-55756-2_12.

[58] F. Liers, T. Nieberg, and G. Pardella, "Via minimization in vlsi chip design application of a planar max-cut algorithm," 2011. http://e-archive.informatik.uni-koeln.de/630/.

[59] M. W. Coffey, "Adiabatic quantum computing solution of the knapsack problem," arXiv, 2017. https://doi.org/10.48550/arxiv.1701.05584.

[60] A. Verma and M. Lewis, "Variable reduction for quadratic unconstrained binary optimization," 2021. https://doi.org/10.48550/arxiv.2105.07032.

[61] 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. https://doi.org/10.1007/s11128-021-03226-6.

[62] M. Garey, D. Johnson, and H. So, "An application of graph coloring to printed circuit testing," IEEE Transactions on circuits and systems, vol. 23, no. 10, pp. 591–599, 1976. https://doi.org/10.1109/TCS.1976.1084138.

[63] R. J. Leatherbarrow, "Using linear and non-linear regression to fit biochemical data," Trends in biochemical sciences, vol. 15, no. 12, pp. 455–458, 1990. https://doi.org/10.1016/0968-0004(90)90295-m.

[64] B. Shyti and D. Valera, "The regression model for the statistical analysis of albanian economy," Int. J. Math. Trends Technol, vol. 62, no. 2, pp. 90–96, 2018. https://doi.org/10.14445/22315373/IJMTT-V62P513.

[65] P. Date, D. Arthur, and L. Pusey-Nazzaro, "QUBO formulations for training machine learning models," Scientific Reports, vol. 11, no. 1, pp. 1–10, 2021. https://doi.org/10.1038/s41598-021-89461-4.

[66] L. Giuffrida, "Solving the binary optimization problem for linear regression with grover adaptive search," December 2021. https://webthesis.biblio.polito.it/21108/.

[67] H. M. Waidyasooriya and M. Hariyama, "A gpu-based quantum annealing simulator for fully-connected ising models utilizing spatial and temporal parallelism," IEEE Access, vol. 8, pp. 67929–67939, 2020. https://doi.org/10.1109/ACCESS.2020.2985699.

[68] Y.-H. Chung, C.-J. Shih, and S.-H. Hung, "Accelerating simulated quantum annealing with gpu and tensor cores," in High Performance Computing (A.-L. Varbanescu, A. Bhatele, P. Luszczek, and B. Marc, eds.), (Cham), pp. 174–191, Springer International Publishing, 2022. https://doi.org/10.1007/978-3-031-07312-0_9.

[69] T. Okuyama, M. Hayashi, and M. Yamaoka, "An ising computer based on simulated quantum annealing by path integral monte carlo method," in 2017 IEEE International Conference on Rebooting Computing (ICRC), pp. 1–6, 2017. https://doi.org/10.1109/ICRC.2017.8123652.

[70] H. M. Waidyasooriya, Y. Araki, and M. Hariyama, "Accelerator architecture for simulated quantum annealing based on resource-utilization-aware scheduling and its implementation using opencl," in 2018 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), pp. 335–340, 2018. https://doi.org/10.1109/ISPACS.2018.8923263.

[71] H. M. Waidyasooriya, M. Hariyama, M. J. Miyama, and M. Ohzeki, "Opencl-based design of an fpga accelerator for quantum annealing simulation," The Journal of Supercomputing, vol. 75, no. 8, pp. 5019–5039, 2019. https://doi.org/10.1007/s11227-019-02778-w.

[72] C.-Y. Liu, H. M. Waidyasooriya, and M. Hariyama, "Data-transfer-bottleneck-less architecture for fpga-based quantum annealing simulation," in 2019 Seventh International Symposium on Computing and Networking (CANDAR), pp. 164–170, 2019. https://doi.org/10.1109/CANDAR.2019.00028.

[73] H. M. Waidyasooriya and M. Hariyama, "Highly-parallel fpga accelerator for simulated quantum annealing," IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 4, pp. 2019–2029, 2021. https://doi.org/10.1109/TETC.2019.2957177.

[74] C.-Y. Liu, H. M. Waidyasooriya, and M. Hariyama, "Design space exploration for an fpga-based quantum annealing simulator with interaction-coefficient-generators," The Journal of Supercomputing, vol. 78, no. 1, pp. 1–17, 2022. https://doi.org/10.1007/s11227-021-03859-5.

[75] H. M. Waidyasooriya and M. Hariyama, "Temporal and spatial parallel processing of simulated quantum annealing on a multicore cpu," The Journal of Supercomputing, vol. 78, no. 6, pp. 8733–8750, 2022. https://doi.org/10.1007/s11227-021-04242-0.

[76] E. Forno, A. Acquaviva, Y. Kobayashi, E. Macii, and G. Urgese, "A parallel hardware architecture for quantum annealing algorithm acceleration," in 2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), pp. 31–36, 2018. https://doi.org/10.1109/VLSI-SoC.2018.8644777.

[77] D. Willsch, M. Willsch, F. Jin, K. Michielsen, and H. De Raedt, "Gpu-accelerated simulations of quantum annealing and the quantum approximate optimization algorithm," Computer Physics Communications, vol. 278, p. 108411, 2022. https://doi.org/10.1016/j.cpc.2022.108411.

[78] M. Suzuki, "Relationship between d-dimensional quantal spin systems and (d+ 1)-dimensional ising systems: Equivalence, critical exponents and systematic approximants of the partition function and spin correlations," Progress of theoretical physics, vol. 56, no. 5, pp. 1454–1469, 1976. https://doi.org/10.1143/PTP.56.1454.

[79] D. B. Hitchcock, "A history of the metropolis–hastings algorithm," The American Statistician, vol. 57, no. 4, pp. 254–257, 2003. https://doi.org/10.1198/0003130032413.

[80] D. Venturelli, S. Mandrà, S. Knysh, B. O'Gorman, R. Biswas, and V. Smelyanskiy, "Quantum optimization of fully connected spin glasses," Physical Review X, vol. 5, no. 3, p. 031040, 2015. https://doi.org/10.1103/PhysRevX.5.031040.

[81] R. H. Swendsen and J.-S. Wang, "Replica monte carlo simulation of spin-glasses," Physical review letters, vol. 57, no. 21, p. 2607, 1986. https://doi.org/10.1103/PhysRevLett.57.2607.

[82] D. J. Earl and M. W. Deem, "Parallel tempering: Theory, applications, and new perspectives," Physical Chemistry Chemical Physics, vol. 7, no. 23, pp. 3910–3916, 2005. https://doi.org/10.1039/B509983H.

[83] M. Sambridge, "A Parallel Tempering algorithm for probabilistic sampling and multimodal optimization," Geophysical Journal International, vol. 196, pp. 357–374, 10 2013. https://doi.org/10.1093/gji/ggt342.

[84] K. Hukushima and Y. Iba, "Population annealing and its application to a spin glass," in AIP Conference Proceedings, vol. 690, pp. 200–206, American Institute of Physics, 2003. https://doi.org/10.1063/1.1632130.

[85] W. Wang, J. Machta, and H. G. Katzgraber, "Population annealing: Theory and application in spin glasses," Physical Review E, vol. 92, no. 6, p. 063307, 2015. https://doi.org/10.1103/physreve.92.063307.

[86] M. Weigel, L. Y. Barash, M. Borovskỳ, W. Janke, and L. N. Shchur, "Population annealing: Massively parallel simulations in statistical physics," in Journal of Physics: Conference Series, vol. 921, p. 012017, IOP Publishing, 2017. https://doi.org/10.1088/1742-6596/921/1/012017.

[87] A. E. Eiben and C. A. Schippers, "On evolutionary exploration and exploitation," Fundamenta Informaticae, vol. 35, no. 1-4, pp. 35–50, 1998. https://doi.org/10.3233/FI-1998-35123403.

[88] Microsoft, "Population annealing." [ON-LINE] https://learn.microsoft.com/en-us/azure/quantum/optimization-population-annealing.

[89] J. Machta, "Population annealing with weighted averages: A monte carlo method for rough free-energy landscapes," Physical Review E, vol. 82, no. 2, p. 026704, 2010. https://doi.org/10.1103/PhysRevE.82.026704.

[90] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, "What is the computational value of finite-range tunneling?," Physical Review X, vol. 6, no. 3, p. 031015, 2016. https://doi.org/10.1103/PhysRevX.6.031015.

[91] M. Le Bellac, Quantum physics. Cambridge University Press, 2011.

[92] H. Irie, H. Liang, S. Gongyo, T. Hatsuda, et al., "Hybrid quantum annealing via molecular dynamics," Scientific reports, vol. 11, no. 1, pp. 1–9, 2021. https://doi.org/10.1038/s41598-021-87676-z.

[93] Dwave, "dwave-hybrid." [ON-LINE] https://github.com/dwavesystems/dwave-hybrid.

[94] "D-wave hybrid solver service: An overview." https://www.dwavesys.com/media/4bnpi53x/14-1039a-b_dwave_hybrid_solver_service_an_overview.pdf.

[95] "Intel Xeon Gold 6134 processor - product specification." [Online] https://ark.intel.com/content/www/us/en/ark/products/120493/intel-xeon-gold-6134-processor-24-75m-cache-3-20-ghz.html, accessed 25-October-2021.

[96] I. Hen, J. Job, T. Albash, T. F. Rønnow, M. Troyer, and D. A. Lidar, "Probing for quantum speedup in spin-glass problems with planted solutions," Physical Review A, vol. 92, no. 4, p. 042325, 2015. https://doi.org/10.1103/PhysRevA.92.042325.

[97] S. Mandra, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, "Strengths and weaknesses of weak-strong cluster problems: A detailed overview of state-of-the-art classical heuristics versus quantum

approaches," Physical Review A, vol. 94, no. 2, p. 022337, 2016. https://doi.org/10.1103/PhysRevA.94.022337.
[98] T. Albash and D. A. Lidar, "Demonstration of a scaling advantage for a quantum annealer over simulated annealing," Physical Review X, vol. 8, no. 3, p. 031016, 2018. https://doi.org/10.1103/PhysRevX.8.031016.
[99] M. Kowalsky, T. Albash, I. Hen, and D. A. Lidar, "3-regular three-xorsat planted solutions benchmark of classical and quantum heuristic optimizers," Quantum Science and Technology, vol. 7, no. 2, p. 025008, 2022. https://doi.org/10.1088/2058-9565/ac4d1b.
[100] M. R. Zielewski and H. Takizawa, "A method for reducing time-to-solution in quantum annealing through pausing," in International Conference on High Performance Computing in Asia-Pacific Region, pp. 137–145, 2022. https://doi.org/10.1145/3492805.3492815.
[101] A. Moraglio, S. Georgescu, and P. Sadowski, "AutoQubo: data-driven automatic QUBO generation," in Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 2232–2239, 2022. https://doi.org/10.1145/3520304.3533965.

• • •

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 (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).