


Quantum pricing-based column-generation framework for hard combinatorial problems

Wesley da Silva Coelho ^{*}, Loïc Henriët, and Louis-Paul Henry
PASQAL SAS, 7 Rue Léonard de Vinci, 91300 Massy, France

 (Received 13 January 2023; accepted 17 March 2023; published 30 March 2023)

In this work we present a complete hybrid classical-quantum algorithm involving a quantum sampler based on neutral-atom platforms. This approach is inspired by classical column-generation frameworks developed in the field of operations research and shows how quantum procedures can assist classical solvers in addressing hard combinatorial problems. We benchmark our method on the minimum vertex coloring problem and show that the proposed hybrid quantum-classical column-generation algorithm can yield good solutions in relatively few iterations. We compare our results with state-of-the-art classical and quantum approaches.

DOI: [10.1103/PhysRevA.107.032426](https://doi.org/10.1103/PhysRevA.107.032426)

I. INTRODUCTION

Combinatorial optimization is at the heart of many real-world problems. It consists in finding the “best” out of a finite, but prohibitively large, set of options. Column generation [1] is an iterative method that was developed to solve this kind of difficult mathematical problem, such as linear formulations where the problem may be too large to consider all options explicitly. In this method, variables are associated with each option. The algorithm starts by solving the considered problem with a limited set of variables (or options), known as the restricted master problem (RMP), and then iteratively adds variables to improve the objective function. The generation of new variables is done by an algorithm specifically tailored to this task: During each iteration, the related new subproblem to be solved, usually referred to as a pricing subproblem (PSP), relies on the duality theory [2] to provide new variables, if any exist, only if they can improve the current solution of the restricted master problem. The iterative process stops when new variables cease to improve the objective function, which is proven mathematically. However, solving the pricing subproblems usually represents the bottleneck of the column-generation approach as it amounts to solving several simpler, but still hard, optimization problems. Hence, designing an efficient way to solve the pricing subproblems is the most important step to ensure high-quality solutions to the RMP while minimizing time and resource consumption.

In recent years, both academic and industrial communities have been putting a great deal of effort into designing quantum hardware and algorithms that could provide a real advantage over classical computers. As pointed out in [3], this advantage can take the form of more accurate results, a faster convergence, or even lower-energy consumption. Such quantum algorithms can be used along with state-of-the-art classical solutions, such as the column-generation algorithm, to create powerful hybrid classical-quantum frameworks. A

wide spectrum of quantum computing platforms is currently being developed, using different kinds of two-level systems as qubits, including Josephson junctions [4,5], trapped ions [6,7], photons [8], or neutral atoms [9,10]. Each of these allows for different quantum processing unit (QPUs) architectures, with their own advantages and limitations when it comes to the connectivity of the qubits or the types of operations that are easily implemented. Extensive knowledge of these platforms allows for the development of hardware-efficient approaches, designed specifically for each of them. In particular, this allows for identifying the classical bottlenecks that are best suited for being replaced by a quantum approach.

In this work we propose a complete hybrid classical-quantum column-generation framework whose pricing subproblems can be efficiently solved in neutral-atom-based QPUs. Unlike other hybrid approaches such as the quantum approximate optimization algorithm (QAOA), the core part of the resolution is here carried out by a classical solver and the quantum processing unit is used as a sampler to restrict the search space. Requiring only $|V|$ qubits for a given graph $\mathcal{G} = (V, \mathcal{E})$, the related pricing subproblems are then solved by a neutral-atom-based sampler specifically tailored to improve the current solution of the associated master problem. Compared to classical and quantum greedy approaches, we show numerically that the proposed hybrid column generation can improve significantly the quality of the solutions while reducing the number of iterations on the quantum device. Finally, by taking advantage of some quantum features, e.g., state superposition, we find that the hybrid column-generation method returns the (near-)optimal solution faster than the classical one, i.e., where no QPU is involved. Figure 1 summarizes our proposed approach.

This paper is structured as follows. We first introduce the main aspects of the graph theory and the related combinatorial problems in Sec. II. After reviewing related works in Sec. III, we give a brief introduction to neutral-atom-based quantum computing in Sec. IV. Section V introduces the main idea of the column-generation approach. In Sec. VI we present in depth our proposed hybrid classical-quantum approach to solving the minimum vertex coloring problem (MVCP), while

^{*}wesley.coelho@pasqal.com

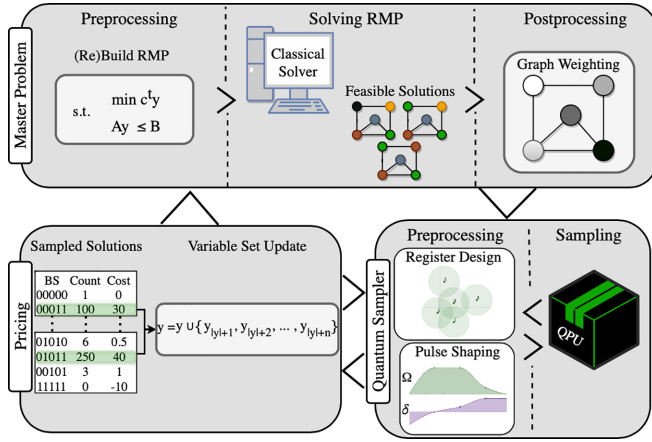


FIG. 1. Workflow of the hybrid classical-quantum column-generation approach. First, a minimal subset y of variables is generated in such a way that it ensures a feasible solution for the reduced master problem (e.g., with only the singletons of the graph). The RMP is then solved by a classical solver. The next steps are related to the pricing subproblems, which are solved by considering the dual values from the solved RMP in order to find more variables that can potentially improve the current solution of the RMP. If such variables exist, then the RMP is updated with the new variables and is solved again. The search for new variables is done by a quantum sampler specifically tailored to consider different inputs related to each pricing iteration. These last steps are repeated until no column is generated by the PSP.

the results of our numerical experiments are discussed in Sec. VII.

II. BACKGROUND

Combinatorial problems [11] have been extensively studied by both academic and industrial communities and have a vast range of applications in real-world systems. Those problems can naturally be defined on graphs, which are data structures composed of a set of elements called vertices (also known as nodes) that can potentially be connected. These connections are called edges and might potentially encode different information such as the importance of such connections (as weights) or the distance between their end points. Similarly, different labels and weights can also be associated with vertices in order to differentiate them. Formally, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is composed of a set of vertices \mathcal{V} and edges $\mathcal{E} \in \mathcal{V}^2$ representing the existence of a connection between vertices u and v from \mathcal{V} .

Several real-world optimization problems, from a vast spectrum of fields, can be mapped to graph problems. For instance, graphs can be used to encode social experiments [12], telecommunication networks [13], and physical systems [14]. The related optimization problems typically consist in selecting a subset of vertices and/or edges optimally satisfying certain rules. This kind of discrete optimization problem is highly relevant for quantum computing (QC), particularly in the case of noisy intermediate-scale quantum-era platforms [15,16]. In that case, results are typically obtained via repeated measurements of the final state of the system. The solutions

are then inferred by the selection of the best sampled state through computationally cheap classical postprocessing.

In the case of neutral-atom QPUs, the spatial arrangement of qubits can be made such that the Ising Hamiltonian describing the interactions in the system is closely related to a given cost function to be minimized. This is what makes this platform notably well suited to solving graph combinatorial problems [10,17–19]. As the state of the computational basis in which the qubits are measured has a direct correspondence to the solution to the graph problem, this type of QC is particularly robust to noise (noise can even be an advantage [20]). For instance, maximum independent set [21] and maximum cut [3] problems can be efficiently solved by approaching the ground state of the quantum system with adiabatic annealing [17,22] or similar methods. In the following, we formally defined some graph problems and show how they can be solved by quantum-based approaches.

A. Combinatorial problems

In the following, we present the maximum independent set, which is a fundamental part of the proof of concept of our hybrid approach. Then we formally define the vertex coloring problem and present the related mathematical formulation.

1. Maximum independent set problem

An independent set in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a subset of vertices $\tilde{\mathcal{V}} \subset \mathcal{V}$ such that no pair of elements from $\tilde{\mathcal{V}}$ is connected by an edge. The independent sets of \mathcal{G} can formally be defined as

$$\mathcal{I}_{\mathcal{G}} = \{\tilde{\mathcal{V}} \subset \mathcal{V} \mid \tilde{\mathcal{V}}^2 \cap \mathcal{E} = \emptyset\}, \quad (1)$$

where $\tilde{\mathcal{V}}^2$ are all the possible edges connecting the vertices in $\tilde{\mathcal{V}}$. Therefore, the maximum independent set (MIS) \mathcal{M} is the largest set of $\mathcal{I}_{\mathcal{G}}$:

$$\mathcal{M}(\mathcal{G}) = \operatorname{argmax}_{\tilde{\mathcal{V}} \in \mathcal{I}_{\mathcal{G}}} |\tilde{\mathcal{V}}|. \quad (2)$$

The MIS problems can be alternatively described in terms of their quadratic unconstrained binary optimization (QUBO) formulations. Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Let x_u be a binary variable associated with each vertex $u \in \mathcal{V}$ that is 1 if vertex u is selected to be in the independent set and 0 otherwise. Hence, the binary vector $\mathbf{x} = \{x_1, \dots, x_{|\mathcal{V}|}\}$ can be put in one-to-one correspondence with partitions $\tilde{\mathcal{V}}$ of the vertex set \mathcal{V} via the identification

$$\tilde{\mathcal{V}}(\mathbf{x}) = \{u \in \mathcal{V} \mid x_u = 1\}. \quad (3)$$

The solution to the maximum (weighted) independent set problem is then given by

$$\mathcal{M}(\mathcal{G}) = \operatorname{argmin}_{\mathbf{x} \in \{0,1\}^{|\mathcal{V}|}} \left(- \sum_{u \in \mathcal{V}} w_u x_u + \alpha \sum_{\{u,v\} \in \mathcal{E}} x_u x_v \right), \quad (4)$$

where the parameter w_u represents the weight associated with each vertex $u \in \mathcal{V}$, while $\alpha > 0$ is an arbitrary coefficient to penalize unfeasible solutions. Note that, on unweighted graphs, all w parameters are set to 1, while the penalty coefficient α must hold any value equal to or greater than 2.

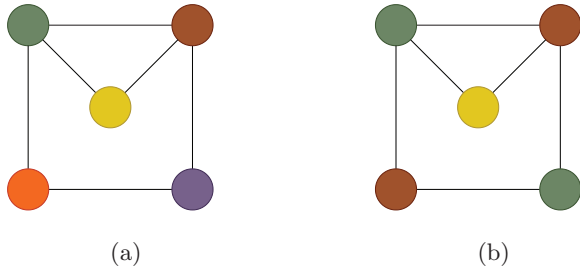


FIG. 2. Two coloring solutions to the same graph with five vertices, six edges, and $\mathcal{X}(\mathcal{G}) = 3$. The set C has five available colors: green, brown, orange, purple, and yellow. (a) Trivial coloring, where each color is mapped to exactly one vertex. (b) Optimal solution for the same instance.

2. Minimum vertex coloring problem

The vertex coloring problem has several applications in real-world optimization problems such as network design [23] and task scheduling [24]. A vertex coloring is an assignment of colors (or labels) to each vertex of a graph such that any two identically colored vertices are not connected by an edge. The minimum vertex coloring problem consists then in finding a feasible coloring while minimizing the number of colors (or labels) assigned; the minimum number of colors used to color all vertices of a given graph \mathcal{G} is called its chromatic number, hereafter denoted by $\mathcal{X}(\mathcal{G})$. The minimum vertex coloring can be formally defined as follows.

Definition 1. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with a set $V = \{u_1, \dots, u_{|\mathcal{V}|}\}$ of vertices and a set $\mathcal{E} \subset \mathcal{V}^2$ of edges. Also, let C be a set of available colors. The minimum vertex coloring problem consists in coloring each vertex of \mathcal{G} with exactly one color from C in a such way that the number of used colors is minimized while ensuring that no two adjacent vertices have the same color.

Figure 2 shows two possible coloring solutions for the same graph. By applying a trivial coloring [see Fig. 2(a)], each color is mapped to exactly one vertex; this simple approach always gives a feasible solution to the problem. As shown in Fig. 2(b), however, the related chromatic number, i.e., the optimal solution, can be reduced to 3. It is worth noting that, given a feasible solution for the vertex coloring problem, any subset of vertices colored with the same color is also an independent set. Hence, finding the minimum subset of independent sets that cover all vertices of a given graph \mathcal{G} is equivalent to solving the MVCP in the same graph. Finding the chromatic number of a graph, however, is one of Karp’s 21 NP-complete problems [25].

B. Extended formulation for the minimum vertex coloring problem

We now present an extended formulation¹ for the minimum vertex coloring problem, which is used within our proposed hybrid approach. First, let S be a set of all possible independent sets in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Also, let b_{us} be a binary

parameter that is 1 if the vertex $u \in V$ is present in the independent set $s \in S$ and 0 otherwise. Finally, we associate a binary variable y_s with each independent set $s \in S$; it is 1 if the related independent set is selected and 0 otherwise. Solving the MVCP then amounts to solving the extended formulation

$$\min \sum_{s \in S} y_s \tag{5}$$

such that

$$\sum_{s \in S} b_{us} y_s = 1 \forall u \in \mathcal{V}, \tag{6}$$

$$y_s \in \{0, 1\} \forall s \in S, \tag{7}$$

where (5) is set to minimize the number of selected independent sets while ensuring that each vertex of the graph is present in exactly one of them [see Eq. (7)]. Note that, by considering each independent set $s \in S$ as a color assignment, the adjacency constraints related to the minimum vertex coloring problem are automatically respected [see the definition (1)].

The number of all independent sets in a graph, and hence the number of y_s variables, can be extremely large, exponentially growing as the number of vertices in the graph increases. Hence, as one may anticipate, finding all such sets on a given graph is a very hard task and can be very time and resource consuming even for small instances. To overcome the aforementioned limitations, we propose a hybrid classical-quantum column-generation-based framework to efficiently solve the proposed extended formulation by enumerating only a small subset of independent sets. In what follows, we present related works and discuss how a quantum sampler can be integrated into classical frameworks.

III. RELATED WORK

Several quantum algorithms have been proposed for solving graph coloring problems and most of them rely on a quantum-annealing-based approach. In [26] Kudo investigated a real-time quantum-dynamics-based quantum-annealing approach where the related Hamiltonian is designed to naturally respect all problem-related constraints without adding penalty terms. Ardelean and Udrescu [27], on the other hand, proposed a genetic algorithm-based quantum approach to solve both vertex and edge coloring problems in different highly configurable circuit-based models. Titiloye and Crispin [28] compared classical and quantum-annealing approaches in solving graph coloring problems. According to them, the path-integral Monte Carlo-based quantum-annealing (QA) algorithm outperforms its classical counterpart.

Silva *et al.* [29] proposed another approach in which the problem-related set of constraints is transformed into an energy minimization problem to output a QUBO formulation, which is then solved in a quantum-annealer platform. However, by running various numerical simulations and comparing results obtained with standard and enhanced circuit-based QAOAs, Tabi *et al.* [30] indicated the limitation of the existing QA hardware solutions for solving the vertex coloring problem. Also, Kwok and Pudenz [31] compared simulated and quantum-annealing approaches for solving the proposed QUBO formulation for the vertex coloring problem. Using D-Wave 2X as an independent set sampler for a simple greedy

¹Extended formulations are mathematical models in which the number of variables grows exponentially as the input increases.

framework, the authors showed that the proposed quantum sampler could improve the results with high probability on small graphs due to hardware limitations.

Fabrikant and Hogg [32] introduced a quantum heuristic for graph coloring for instances that can be solved with at most three colors. Using two qubits for each vertex of the graph, an approximate asymptotic analysis suggests polynomial-time cost for solving the related three-coloring problem. Moreover, Shimizu and Mori [33] introduced an exponential-space quantum algorithm to solve the MVCP in $O(1.9140^{|\mathcal{V}|})$ running time. They proposed a quantum random access memory framework based on the quantum dynamic programming of Ambainis *et al.* [34] with applications of Grover's search to branching algorithms. Moreover, Vitali *et al.* [35] proposed a greedy quantum algorithm for solving the MVCP by iteratively computing the solutions of maximum independent set problems. By simulating a framework on a classical computer to reproduce the Rydberg blockade phenomenon on neutral-atom-based QPUs, they showed that their approach can always find feasible solutions for the problem. More details about their approach are presented in Appendix A.

Finally, Ossorio-Castillo and Pena-Brage [36] proposed a quantum-annealing-based method to solve one of the two pricing subproblems of a column-generation-based approach for the refinery scheduling problem. They first decomposed the problem into one master problem and two pricing subproblems and formulated one of them as a QUBO model. While only the QUBO-related pricing subproblem is solved using a quantum-annealing approach, the master and the other pricing subproblem are solved with a classical solver. The quantum system is based on D-Wave's quantum annealers and is designed to return only the optimal solution. Due to some serious limitations related to either the problem size or the connectivity of its variables, the authors could guarantee high-quality solutions for only a few instances of small graphs.

Note that, even though those works proposed interesting approaches to solving combinatorial problems (sometimes only the decision version), little attention has been given to hybrid and analog approaches, especially using neutral-atom-based QPUs. In what follows, we introduce a complete quantum pricing framework based on neutral-atom QPUs that can be easily embedded into a column-generation algorithm to solve hard combinatorial problems, such as graph coloring problems.

IV. NEUTRAL-ATOM QPUS

In neutral-atom-based QPUs, lasers or microwaves are used to induce transitions between electronic states of the valence electron of alkali-metal (typically rubidium) atoms. Different pairs of electronic levels can be used as qubits. Here we will solely focus on the case where those two states are the electronic ground state $|g\rangle \equiv |0\rangle$ and an s Rydberg level $|r\rangle \equiv |1\rangle$. In that case, atoms can be placed arbitrarily in space, so the effective Hamiltonian of the atoms at time t can be written as

$$H(t) = \Omega(t) \sum_{u=1}^{|\mathcal{V}|} \hat{\sigma}_u^x - \Delta(t) \sum_{u=1}^{|\mathcal{V}|} \hat{n}_u + \sum_{u<v=1}^{|\mathcal{V}|} U_{uv} \hat{n}_u \hat{n}_v, \quad (8)$$

where the amplitude (giving the Rabi frequency) $\Omega(t)$ and detuning $\Delta(t)$ of the laser can be controlled over time and the interaction strength $U_{uv} \propto |\mathbf{r}_u - \mathbf{r}_v|^{-6}$ is a function of the distance between atom u and atom v . Throughout this paper, we set $\hbar = 1$. Note that in the present work we consider only a uniform global laser control over the atoms.

A key property of Rydberg physics is the so-called Rydberg blockade mechanism [9]: The two-body interaction term in (8) forbids the simultaneous excitation of two atoms that are closer than a certain distance. Given a set of atoms and their positions, one can then define a graph such that each atom corresponds to a vertex and in such a way that two vertices are connected by an edge if and only if the distance between the related atoms is shorter than a given threshold. This kind of graph is known as a unit-disk (UD) graph and it is the most natural graph to encode in a neutral-atom QPU. For this graph class, one can ensure that the evolution of the quantum system is restricted to a subspace of the complete Hilbert space corresponding to independent sets of the graph. By setting $\Omega(t) = 0$ and adjusting the value of Δ , one can ensure that the ground state corresponds to an MIS. Because of these properties, people have explored quantum annealing as a way to solve optimization problems on graphs [37]. For non-UD graphs, however, one can construct alternative approaches, similar to what is done in the QAOA or variational quantum eigensolvers [38].

V. PROBLEM DECOMPOSITION

We dedicate this section to fully describing the decomposition of the proposed extended formulation (5)–(7), a fundamental step to solve the related combinatorial problem with a column-generation-based algorithm. The need to apply such a mathematical strategy comes from the fact that, in most of cases, generating all elements that will be related to the variables of an extended formulation is a very hard task. For instance, enumerating all independent sets of a graph can be impractical even for small instances. To overcome the aforementioned issue, we decompose the problem under consideration into two problems, the RMP and PSP. While the former has only a small subset of variables needed to find a solution to the problem, the latter is designed to provide new elements, e.g., independent sets, that respect all technical constraints imposed by the RMP. These new elements are then added as new variables (also seen as columns) to the mathematical model related to the RMP as they might potentially improve the quality of the solution, e.g., decreasing the number of colors needed to solve the vertex coloring problem.

Column-generation-based approaches rely on the duality theory [39], which states that optimization problems can be addressed from two different perspectives: the primal problem or its counterpart, the dual problem. The relationship between these two problems is the following: (i) For each variable (constraint) in the primal problem, there is a related constraint (variable) in the dual problem and (ii) the optimization direction, e.g., maximization or minimization, on the dual is inversely related to its primal counterpart. For each suboptimal solution that satisfies all the constraints on the primal problem, there is at least one direction to move in such a way that the objective function is improved. Such improving

directions are represented by the vector of dual variables and optimizing them is equivalent to tightening the bounds of the primal problem. For an in-depth discussion on the primal-dual relationship, one may refer to [39]. After solving a linear model related to a primal problem, one can easily get such a direction vector, i.e., dual variables, by calling some built-in function proposed by the solver used in the process.

In order to design an efficient column-generation framework, the PSP is then formulated in such a way to incorporate the dual information provided by current solutions of the RMP, which implies solving a different pricing instance each time the subproblem is called. Hence, by applying the duality theory, the PSP searches for new variables that can improve the objective function of the RMP, and once it is mathematically proven that it is no longer possible to generate such variables, i.e., new columns, the loop-based procedure stops. This approach is very powerful when only a few variables are normally activated, i.e., taking any value other than zero, in the optimal solution to a given combinatorial problem. Hence, applying such a technique, only a very small subset of variables needs to be generated by the pricing routine.

In what follows, we present a decomposition scheme for solving the proposed extended formulation for the MVCP. The main idea relies on generating a restricted model with only a subset of independent sets and iteratively updating it with new variables, i.e., columns, that have the potential to improve the current solution.

A. Restricted master problem

Since \mathcal{G} has an exponential number of potentially suitable colorings represented by the related set S of independent sets, the extended formulation (5)–(7) admits an exponential number of variables. To overcome the difficulty of generating S , we propose to generate only a small subset $S' \subseteq S$ of variables that are needed to solve the master problem. For instance, a trivial solution might be initializing S' with only the singletons of the input graph. The reduced model is then hereafter referred to as the restricted master problem and can be defined as

$$\min \sum_{s \in S'} y_s \tag{9}$$

such that

$$\sum_{s \in S'} b_{us} y_s = 1 \forall u \in \mathcal{V}, \tag{10}$$

$$0 \leq y_s \leq 1 \forall s \in S'. \tag{11}$$

Note that, in order to apply the duality theory, we must solve the linear relaxation on the RMP, meaning the y variables are no longer binary in this formulation. The RMP is then solved again with the integrality constraints (7) once it has all variables needed to provide the optimal solution for the relaxed RPM, which can be proven mathematically; we provide this proof in the following section.

B. Pricing subproblems

We present first the dual formulation related to the relaxed RMP (9)–(11). By associating a dual variable w_u with each

constraint in (10), we define the dual problem as

$$\max \sum_{u \in \mathcal{V}} w_u \tag{12}$$

such that

$$\sum_{u \in \mathcal{V}} w_u b_{us} \leq 1 \forall s \in S', \tag{13}$$

$$w_u \in \mathcal{R} \forall u \in \mathcal{V}. \tag{14}$$

The separation of inequalities (13) represents the pricing subproblems related to the extended formulation (9)–(11). The PSP consists then in finding a new coloring set in such a way that all adjacency constraints would be respected while improving the solution cost of the RMP, i.e., decreasing the value of the objective function (9). For this purpose, let \bar{w}_u be the components of the current dual solution of the RMP related to constraints (10). By setting each dual variable \bar{w}_u as the weight of the related vertex $u \in \mathcal{V}$, finding a new independent set (IS) under such conditions amounts to solving the maximum weighted independent set (MWIS) formulation

$$\max \sum_{u \in \mathcal{V}} \bar{w}_u x_u \tag{15}$$

such that

$$x_u + x_v \leq 1 \forall (u, v) \in \mathcal{E}, \tag{16}$$

$$x_u \in \{0, 1\} \forall u \in \mathcal{V}, \tag{17}$$

where x_u is a binary variable that is 1 if it is in the independent set and 0 otherwise. The inequality (16) ensures that the new independent set respects the adjacency constraints. Note that the pricing formulation (15)–(17) is an integer programming (IP) version of the QUBO formulation (4).

The net gain of adding a new variable related to a solution to the pricing problem is given by the reduced cost. Based on the separation of inequalities (13), we calculate the reduced cost r_s for any independent set s given by solving the formulation (15)–(17) as follows:

$$r_s = 1 - \sum_{u \in \mathcal{V}} \bar{w}_u x_u. \tag{18}$$

Since we minimize the master problem in the work, any solution with a negative reduced cost might potentially improve the solution of the RMP. The pricing problem consists then in finding a new independent set s whose total weight is strictly greater than 1; the total weight of any independent set is calculated as in the cost function (15). Hence, any solution to the formulation (15)–(17) whose cost is greater than 1 can therefore be added to the subset S' ; if such a solution does not exist, then the solution of the RMP cannot be improved and hence the optimal solution the relaxed RPM can be achieved with the current variables related to the subset S' .

VI. SOLVING METHOD

As previously discussed, solving the pricing subproblems is usually the bottleneck in column-generation-based algorithms since it amounts to solving different instances of a hard combinatorial problem multiple times. To overcome this problem, we propose a quantum pricing algorithm that can

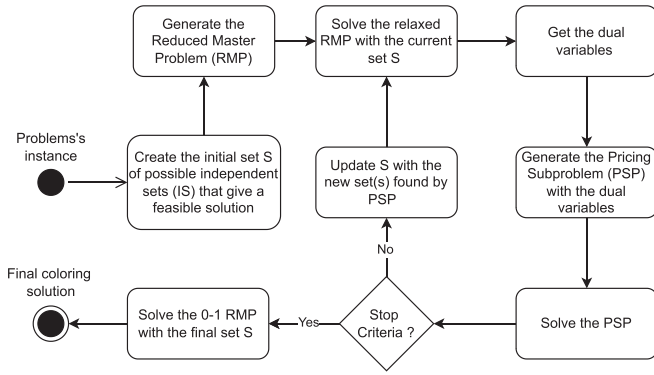


FIG. 3. Interaction between the restricted master problem and the subproblem.

find the (near-)optimal solution faster than the classical one, i.e., where no QPU is involved. For this purpose, let us now describe the column-generation-based framework proposed to solve the minimum vertex coloring problem, which is summarized in Fig. 3.

First, a minimal subset $S' \subseteq S$ of independent sets is generated in such a way that it ensures a feasible solution for the extended formulation (5)–(7). As previously discussed, the most trivial way to build the initial set S' of independent sets is generating only the singletons in the graph; this simple approach always provides a solution for the RMP.

The classical part of the proposed hybrid approach is related to the restricted master. Once the initial set S' is created, the RMP is built and then solved in its linear relaxation form [see the formulation (9)–(11)] by a classical solver, e.g., the GNU linear programming kit (GLPK). The values of the dual variables are also given by the classical solver by running a built routine after solving each version of the RMPs, i.e., with different subsets of variables.

The next steps are related to the pricing subproblems, in which the PSP is solved by applying the values of the related dual variables from the solved RMP. As previously discussed, this step amounts to finding independent sets whose weight is strictly greater than 1. If such elements exist, then they are added to S' . As we detail in the following, we propose a quantum sampler that is specifically tailored to output multiple independent sets under the aforementioned conditions. For each new independent set found by solving the related pricing subproblem, a new variable is created and added to the subset S' . Then the RMP is solved again with the new columns, i.e., independent sets converted into variables. These last steps are repeated until no column is generated by the PSP. Finally, the final RMP is solved with all generated variables, i.e., independent sets, with the integrality constraints (7), as previously discussed.

A. Worked example

Table I shows a worked example of applying the column-generation framework on the graph represented in Fig. 4, where the RMP is solved classically by an IP solver. The first column shows how many PSPs are solved before reaching the final solution. The second column depicts the independent sets selected as the solution for the RMP formulation (9)–(11).

TABLE I. Worked example of applying the column-generation framework on the graph represented in Fig. 4. Only five independent sets (out of ten) are generated to find the optimal solution.

Iteration	RMP solution	Dual solution	MWIS
1	[1], [2], [3], [4], [5]	[1.0, 1.0, 1.0, 1.0, 1.0]	[1, 2, 4]
2	[3], [5], [1, 2, 4]	[1.0, 1.0, 1.0, -1.0, 1.0]	[2, 3, 5]
3	[3], [5], [1, 2, 4]	[1.0, -1.0, 1.0, 1.0, 1.0]	[1, 4]
4	[1, 4], [2, 3, 5]	[1.0, -1.0, 1.0, 0.0, 1.0]	[3, 5]
5	[3, 5], [1, 2, 4]	[1.0, 0.0, 1.0, 0.0, 0.0]	none

The third column presents the dual solution given by the variables w [see the dual formulation (12)–(14)]. Finally, the last column depicts the maximum weighted independent sets generated after running each PSP, where the \bar{w} parameters in the cost function of the formulation (15)–(17) are set to the values of dual variables w ; the generated weighted independent set is then added to the set S' before rerunning the RMP. The final coloring is represented by the last solution for the RMP, where one color is given for each selected independent set.

In this example, we generate all five singletons of the graph as the first subset S' of independent sets before solving the first version of the RMP. As observed, only four more independent sets (out of the remaining ten to be generated) are needed to find the best solution. Indeed, even though the closed loop could be stopped after iteration 4, the optimality is only proven after the fifth iteration, when no independent set whose total weight is greater than 1 can be generated (see the dual solution on the last row, which is used as vertex weights). Note that the independent set generated in the i th iteration is likely to be selected in the solutions of the RPM in the following iteration.

The values of the dual variables can be calculated by solving the dual formulation (12)–(14), where the cost function is set to be equal to the solution cost of the RMP in the same interaction. In this example, the cost function of the dual formulation is set equal to 5, 3, 3, 2, and 2 in the first, second, third, fourth, and fifth iterations, respectively. However, most commercial solvers solve the dual formulation in parallel in order to prove the optimality of the (RMP) primal solution. Hence, the final value of the dual variables can then be easily provided by a built-in solver’s function, e.g., `solver.get_dual()`. Note, however, that only one independent set (the most weighted one) is generated by solving the related pricing formulation (15)–(17) and, due the deterministic nature of this approach, the final solution is always the

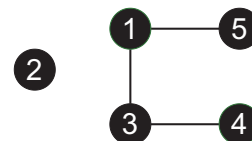


FIG. 4. Illustration of a graph with five vertices and three edges; the set S of all independent sets is composed of the five singletons and the subsets [1, 2], [1, 4], [2, 3], [2, 4], [2, 5], [3, 5], [4, 5], [1, 2, 4], [2, 3, 5], and [2, 4, 5].

TABLE II. Worked example of applying a quantum sampler to solve the pricing subproblems related to the column-generation framework. Considering the graph represented in Fig. 4, only three different pricing subproblems were solved before reaching the final solution.

Iteration	RMP solution	Dual solution	IS generated
1	[1], [2], [3], [4], [5]	[1.0, 1.0, 1.0, 1.0, 1.0]	[1, 2, 4], [2, 4, 5] [2, 3, 5], [1, 2] [2, 5], [2, 3] [4, 5], [3, 5]
2	[3, 5], [1, 2, 4]	[1.0, -0.5, 1.0, 0.5, 0.0]	[1, 4]
3	[1, 4], [2, 3, 5]	[1.0, 0.0, 1.0, 0.0, 0.0]	none

same if the input, i.e., the graph and vertex weights, does not change.

Now let us exemplify how a quantum sampler could be applied to solve the pricing subproblems within the column-generation framework; Table II shows such a worked example. As presented in the previous worked example depicted in Table I, we consider the graph depicted in Fig. 4 to be an initial subset S' composed only of singletons. As observed, only three iterations are needed to find the final solution. Indeed, sampling more independent sets in each pricing iteration can considerably improve the performance of the column-generation algorithm. In this example, nine out of ten possible independent sets are generated. In fact, due to its stochastic nature, a quantum sampler can efficiently provide multiple sets with the same input and hence speed up the convergence of the RMP to the optimal solution.

In what follows, we present how neutral-atom-based systems can be designed to take into consideration the dual values from the RMP to output multiple weighted independent sets and efficiently solve the related PSPs.

B. Embedding strategy

Embedding random graphs into UD representations has proven to be an NP-hard problem [40]. Also, not all graphs have such a realization. For instance, it is not possible to find a UD representation for any K_{1n} star graph with $n > 6$ on a two-dimensional plane. In order to design a near-optimal register to represent any graph given as an input, we use the embedding strategy presented in [3], where an algorithm based on force-directed principles is used to embed graphs into planes in such a way that two connected (disjoint) vertices are placed close to (far from) each other, with a minimum (maximum) distance between them (from the plane's center). See Appendix B for more details.

Once the initial register is created as previously discussed, it has the same number of atoms as the number of vertices on the initial graph \mathcal{G} . However, solving the pricing subproblem within the column-generation framework might potentially imply finding one or more independent sets on a subgraph \mathcal{G}' ; this subgraph is generated with only the vertices with positive weight, i.e., the positive dual value provided by the solved restricted master problem. The light pattern holding the atoms in place is created via a spatial light modulator, and

Algorithm 1. Vertex-atom remapping.

Input: A graph \mathcal{G}' , a register R , and vertex weights W

Output: The reduced register R

- 1: Let $\tilde{\mathcal{V}}'$ be the list of vertices from \mathcal{G}' sorted in descending order by the weight given by W
- 2: Remove all vertices whose weights are less than or equal to zero from $\tilde{\mathcal{V}}'$
- 3: Map vertex $\tilde{\mathcal{V}}'.first$ to the farthest position from the center of R
- 4: $\tilde{\mathcal{V}}' \leftarrow \tilde{\mathcal{V}}' \setminus \tilde{\mathcal{V}}'.first$
- 5: **while** $\tilde{\mathcal{V}}'$ has vertices **do**
- 6: Map vertex $\tilde{\mathcal{V}}'.first$ to the farthest position from all vertices already mapped to an atom in R
- 7: $\tilde{\mathcal{V}}' \leftarrow \tilde{\mathcal{V}}' \setminus \tilde{\mathcal{V}}'.first$
- 8: **end while**
- 9: Remove all atoms not mapped to any vertex from register R
- 10: **return** register R

determining the right settings for this device demands lengthy calibrations [19].

Therefore, creating a new register by running the Fruchterman-Reingold algorithm with the remaining vertices would be too time consuming. Instead of just removing the atoms mapped to the vertices that are no longer in the subgraph \mathcal{G}' , we propose here a vertex-atom remapping strategy that takes into consideration the vertices' weights (dual values).

Algorithm 1 summarizes the main idea of our proposed approach: It receives a subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, a register R composed of atoms and their positions within the QPU, and a vector W representing the weights for all vertices in \mathcal{V}' . First, let $\tilde{\mathcal{V}}'$ be the list of vertices of \mathcal{G}' sorted in descending order by the weight given by W . Then remove all vertices whose weights are less than or equal to zero from $\tilde{\mathcal{V}}'$ (step 2), map the most weighted vertex to the atom farther from the register's center (step 3), and remove the remapped vertex from $\tilde{\mathcal{V}}'$ (step 4). Then the most weighted vertex in $\tilde{\mathcal{V}}'$ is embedded into the atom position farthest from all atoms already mapped to a vertex in \mathcal{G}' and removed from $\tilde{\mathcal{V}}'$; these steps are done until all vertices from \mathcal{G}' are mapped to an atom in the register R . Finally, as seen in step 9, all atoms not mapped to any vertex are removed from R . Let us recall that, since the proposed remapping strategy is applied to solve the PSPs, the weight vector W is generated with dual values provided by the solution of the current RMP, as previously mentioned. A worked example of applying the proposed algorithm is depicted in Fig. 5.

C. Independent set quantum sampler

We propose an independent set sampler based on current-generation neutral-atom quantum processing units, which is inspired by the quantum adiabatic approach described by [41]. In the latter, the main idea is to slowly evolve the system from an easy-to-prepare ground state to the ground state of the final cost Hamiltonian H_C . By slowly evolving the system, the atoms stay in the instantaneous ground state [42]. Here we only aim at keeping the system *close* to the instantaneous

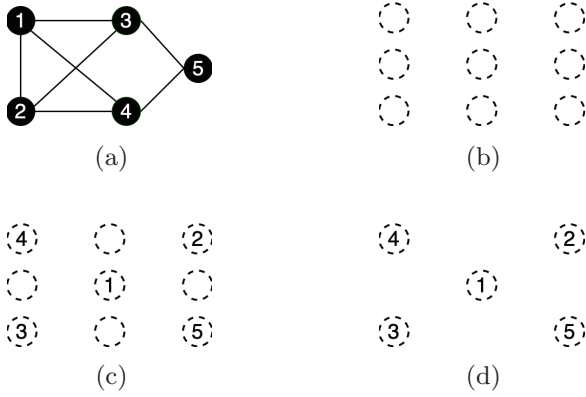


FIG. 5. Vertex-atom mapping for a five-vertex graph on a nine-atom register: (a) graph with five vertices whose weights are to their indices (e.g., the weights of vertices 1 and 5 are equal to 1 and 5, respectively), (b) register with nine atoms in their positions, (c) mapping where the most weighted vertices are far from each other, and (d) final register and the vertices from the subgraph they represent.

ground state, allowing it to pick up components in the low-lying states.

In order to have such an evolution, we continuously vary the detuning $\delta(t)$ and the Rabi frequency $\Omega(t)$ in time, starting with $\Omega(t_0) = 0$ and $\delta(t_0) < 0$ and ending with $\Omega(t_f) = 0$ and $\delta(t_f) > 0$. Hence, the initial ground state of $H(t_0)$ is $|00000\rangle$, while the low-energy space of $H(t_f)$ contains that of the cost Hamiltonian H_C . This protocol originally aims at preparing the ground state and solving the MIS problem. In this work, we use it to sample independent sets whose total weight is equal to or close to that of the MWIS.

To ensure that we are not exciting the system to states that do not form independent sets, we have to estimate the minimal distance between atoms that are disjoint in the graph (this yields Ω_d) and estimate the farthest distance between two connected atoms, which gives Ω_c . For this purpose, let d_{uv} be the distance between nodes u and v from \mathcal{V} , while C_6 is the interaction coefficient related to the quantum device. Then Ω_d and Ω_c are defined as

$$\Omega_d = \operatorname{argmax}_{(u,v) \notin \mathcal{E}} C_6 d_{uv}^{-6}, \quad (19)$$

$$\Omega_c = \operatorname{argmin}_{(u,v) \in \mathcal{E}} C_6 d_{uv}^{-6}, \quad (20)$$

respectively. In UD graphs, keeping $\Omega \in [\Omega_d, \Omega_c]$ ensures that only independent sets appear in the dynamics (provided the dynamics is not too fast) [37,43]. A large value of Ω is desirable to speed up the dynamics of the system and reach states that are *far* from the initial state. That is why we set $\Omega_{\max} = \max(\Omega_c, \Omega_d)$ as the maximum value the Rabi frequency can take during the pulse sequence.

In the case of non-UD graphs, this value of Ω_{\max} is still a good compromise. Indeed, if $\Omega < \Omega_c$ then the effective graph \mathcal{G}' that is encoded in the system contains more edges than the target graph \mathcal{G} . The independent sets of \mathcal{G}' are then not strictly included in the independent sets of \mathcal{G} and the pricing may then miss some variables. On the other hand, if $\Omega > \Omega_d$, all edges of \mathcal{G}' are also edges of \mathcal{G} and therefore the

independent sets of \mathcal{G}' are strictly included in the independent sets of \mathcal{G} . This ensures that the pricing is still able to explore all independent sets and therefore provide new variables to improve the current solution of the RMP.

As inputs, the quantum sampler gets an atom register and the pulse representing the designed Hamiltonian. Then, after running the proposed neutral-atom-based quantum algorithm, different outputs are possible.

- (i) Return only the largest independent set.
- (ii) Return all independent sets.
- (iii) Return all independent sets whose weights are greater than a given threshold.
- (iv) Return only the most weighted independent set.

When applying the two last strategies, one must also provide a weight vector and a cost function as inputs to the quantum sampler. In this work, we apply the third output case mentioned above by applying the cost function (15) to qualify any independent set; note that for unweighted graphs, this cost function can be applied by setting all weights to 1.

D. Solving the pricing subproblems

Let \bar{w} be the vector of dual values related to the solved relaxed RMP as previously discussed. Then we generate the graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \bar{w})$ in such a way that $\mathcal{V}' = \{u \in \mathcal{V} \mid \bar{w}_u > 0\}$ and $\mathcal{E}' = \{(u, v) \in \mathcal{E} \mid \bar{w}_u \bar{w}_v > 0\}$. Also, for each vertex $u \in \mathcal{V}'$, the related weight is given by $\bar{w}_u \in \mathbb{R}$.

We run the quantum algorithm previously described on the related graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \bar{w})$, where \bar{w} is the dual vector from the current solve solution for the primal problem. While we keep only the atoms related to the vertices in \mathcal{V}' , their positions might potentially be permuted as described in Algorithm 1. Also, the pulse shape might be adjusted to the new register by calculating the new Ω_{\max} , which is done by calculating the distance between each pair of qubits, as previously proposed. By applying the objective function (15) to qualify each bit string sampled by the QPU, the S' is then updated with all sampled weighted independent sets whose total weight is strictly greater than 1. Let us recall that the \bar{w} might potentially be a vector composed of very small values, i.e., $\bar{w}_u \ll 1 \forall u \in \mathcal{V}'$, including zero and negative values, meaning that any independent set, including the maximum one, might have a total weight strictly lower than 1; in this case, no independent set is added to S' .

E. Stopping criterion

The algorithm stops the inner loop when no column is generated after solving the related pricing subproblem. This means that either the current subset S' is already composed of all independent sets needed to find the optimal for the relaxed formulation (9)–(11) (if the PSP is exactly solved) or the PSP solver does not find any solution under the imposed constraints (if the PSP is solved heuristically). Once the inner loop stops, the integer linear program (ILP) version of the final RMP is solved with all generated columns, i.e., variables or independent sets, and the integrality constraints (7).

It is important to notice that the final solution given by the related LP formulation (9)–(11) might not be the same (or even the optimal one) for the ILP formulation (5)–(7),

i.e., with the integrality constraints (7). To ensure optimality, the proposed hybrid column generation should be embedded into a branch-and-price framework. This approach, however, is beyond the scope of this work. For more information, one may refer to [1].

VII. NUMERICAL SIMULATIONS

Let us first describe the setup used in our numerical simulations. While random graph instances are generated with the Vladimir-Brandes algorithm [44], which produces Erdős-Rényi graphs, UD-guaranteed graphs are produced as proposed in [45]. It is worth mentioning that random graphs are unlikely to be unit disks. Indeed, the probability of having a UD Erdős-Rényi graph quickly approaches zero as the number of the vertices increases and the density remains stable. For this reason, this graph class, i.e., random graphs, is hereafter referred to as non-UD graphs. For each graph class, we generate 30 instances for three different graph densities by setting the probability p of connecting any pair of vertices with an edge to 20%, 50%, and 80%. Also, the optimal solution of each instance is found by exactly solving the compact formulation of the minimum vertex coloring problem [46].

We compare our proposed hybrid algorithm to several state-of-the-art approaches, both classical and quantum. Indeed, it is possible to solve the pricing problem (15)–(17) with a classical solver, where \mathcal{G}' is given as a vertex-weighted graph. Note, however, that only the optimal weighted independent set is generated during each iteration. As previously discussed, due to the deterministic nature of this approach, hereafter referred to as classical column generation (CG), the final solution does not change if the input, i.e., the graph and vertex weights, remains the same. We also compare our proposed approach to the classical greedy algorithm described in Appendix A, where a maximum independent set is provided by a classical solver in each iteration. We use the GLPK [47] as the linear solver to solve the (reduced) master problem and classical pricing subproblems.

In order to compare our neutral-atom-based quantum sampler to other stochastic approaches, we implement a greedy generator, hereafter referred to as greedy CG, that can randomly generate multiple weighted independent sets. For more details, see Appendix C. We also compare our approach to a simulated annealing (SA)–based solver, hereafter referred to as SA CG, where we minimize the related maximum weighted independent set QUBO matrix described in (4) with the classical D-Wave QUBO sampler [48]; the weights are given by the dual values of the solved RMP in each iteration, while α is set to the sum of absolute values of weights. For all stochastic subroutines, the maximum number of tries is set to 1000. Moreover, several pricing iterations can be done before getting an improvement on the RPM solution. This is due to the inherent symmetry of the solution space related to the instance of the problem, driving the algorithm to generate independent sets with the same cost. In all CG-based approaches, the maximum number of pricing iterations allowed without any improvements in the RMP solution is set to 3. Finally, we also compare our hybrid approach against a quantum version of the greedy algorithm described in Appendix A, where only the largest independent set sampled by the proposed quantum

TABLE III. Noise parameters used in noisy emulations.

SPAM				
Bad preparation η	False positive ϵ	False negative ϵ'	Temperature (μK)	Laser waist (μm)
0.005	0.03	0.08	30	148

sampler described in Sec. VIC is returned in each iteration. This approach is hereafter referred to as quantum greedy and is based on the work proposed by Vitali *et al* [35].

The register related to each pricing subproblem is created by applying the embedding strategy presented in Sec. VIB. To this end, each atom’s position is found with the spring layout function from NETWORKX package [49]; we multiply each position vector by 40 in order to respect the distance constraints imposed by the device. Moreover, we apply the Hamiltonian design strategy described in Sec. VIC.

The evolution of the quantum system under the predicted pulse for a given register is then simulated using PULSER’s simulation module [50]. Both noiseless and noisy simulations are performed. Noiseless simulations involve solving the time-dependent Schrödinger equation. The output of a noiseless simulation is a vector in the Hilbert space that can be sampled a finite number of times in order to mimic a real experimental setup with a limited measurement budget. In order to assess the robustness of our approach against noise, noisy simulations taking into account the current levels of noise in these systems [19,51] are also performed. These calculations are numerically expensive and we restrict our study to state preparation and measurement (SPAM) errors. These are expected to be the main source of noise and can be obtained by postprocessing of noiseless results (for more details, see [38]). We set here the noise parameters to realistic values based on current hardware specifications. They are summarized in Table III. The register and Hamiltonian design are done as previously presented in Secs. VIB and VIC, respectively.

A. Register and pulse redesign strategies

We first analyze the impact of four different registers and pulse, i.e., Hamiltonian, redesign strategies on the performance of our proposed hybrid classical-quantum column-generation approach; while the atom removal (AR) strategy only removes the atoms whose related vertex’s weight is equal to or less than zero, atom index permutation and atom removal (AIPR) strategy also apply the proposed vertex-atom remapping algorithm (see Algorithm 1). In addition to these two strategies, we test the atom removal with Hamiltonian redesign (AR-HDR), in which the new maximal Rabi frequency is recalculated after applying the AR strategy. Finally, in atom index permutation, atom removal, and Hamiltonian redesign (AIPR-HDR) strategy, the new maximal Rabi frequency can also be recalculated after applying the AIPR strategy.

Figure 6 shows the average number of pricing iterations (and the standard deviation with 95% confidence interval) before reaching the stop criteria on different graph classes (UD and non-UD), order (from four up to 14 vertices), and

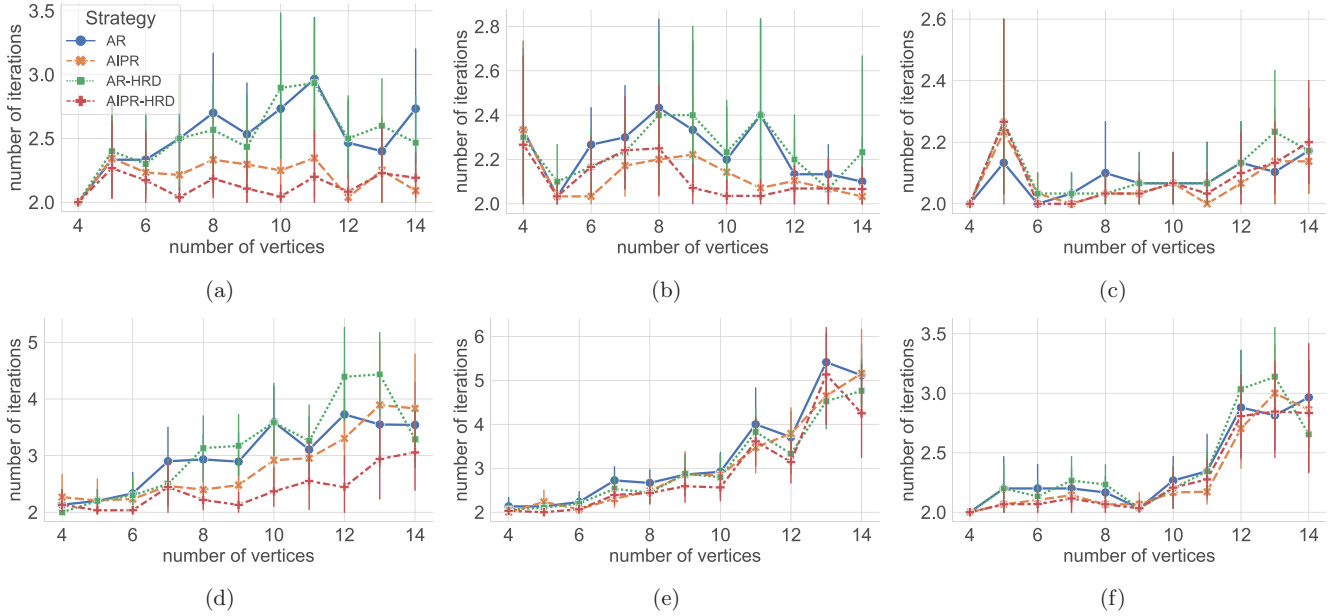


FIG. 6. Number of iterations on the QPU emulator before reaching the stop criteria on different graph classes (UD and non-UD), orders (from four up to 14 vertices), and densities (20%, 50%, and 80% of all possible connections) and applying different register and Hamiltonian redesign strategies: only atom removal (AR), atom index permutation and atom removal (AIPR), atom removal with Hamiltonian redesign (AR-HRD), and AIPR with Hamiltonian redesign (AIPR-HRD). Results are shown on (a)–(c) unit-disk graphs and (d)–(f) non-UD graphs, as well as on (a) and (d) 20%-density graphs, (b) and (e) 50%-density graphs, and (c) and (f) 80%-density graphs.

densities (20%, 50%, and 80% of all possible connections). As observed, only a few calls to the quantum sampler, i.e., pricing iterations, are needed to reach the best solution of the relaxed version² of the master problem related to each graph instance. Indeed, the average number of iterations on non-UD (UD) graphs is always less than 6 (3); as seen in Figs. 6(c) and 6(f), only two (three) sampling processes are done on average to solve UD (non-UD) graphs with four (13) vertices and 80% density when the AIPR (AR-HRD) strategy is applied. Even though non-UD graphs seem to be more difficult to solve, especially those with 50% of density [see Fig. 6(e)], applying different register and pulse redesign strategies could speed up the solving process. While we do not observe any significant impact from redesigning the pulse after only removing useless atoms from the register (see the blue and green lines related to AR and AR-HRD strategies, respectively), recalculating the maximum value of the Rabi frequency after permuting the atoms’ indices, i.e., applying the AIPR-HRD approach, could decrease the number of sampling processes by 44% [see Fig. 6(d)]. Indeed, as seen in Figs. 6(a) and 6(d), this approach has the best overall performance, having a stronger impact on sparse graphs.

B. Quantum and classical approaches

We now compare the number of iterations needed to be run on different graph classes, orders, and densities by applying different approaches: classical CG, greedy CG, SA CG, noise-

less quantum CG, classical greedy, and quantum greedy. We apply the AIPR-HRD strategy for redesigning each pricing subproblem within the quantum CG framework. While this indicator refers to how many times the PSP is solved within both classical and quantum column-generation frameworks for coloring a given graph, it indicates how many independent sets are generated during the while loop in Algorithm2 by using both classical and quantum methods as previously discussed. Figure 7 shows the average number of iterations and the standard deviation (with 95% confidence interval) on 30 graphs randomly generated as presented above.

First, we observe that the quantum greedy approach has the same overall performance as its classical counterpart, showing that our quantum sampler can solve the maximum independent set problem efficiently. Also, both strategies have the same linear behavior related to the size of the graph, i.e., the number of edges it contains, being most impacted by dense graphs [see Figs. 7(c) and 7(f)]. This behavior is expected

Algorithm 2. Greedy algorithm.

```

Input: A graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  and a set  $C$  of available colors
Output: Color vertex assignment
1: Let  $\mathcal{G}' \leftarrow \mathcal{G}$ 
2:  $c \leftarrow 1$ 
3: while  $\mathcal{G}'$  has vertices do
4:   Find a (maximum) independent set  $\mathcal{I}$  in  $\mathcal{G}'$ 
5:   Assign color  $c \in C$  to all vertices from  $\mathcal{I}$  in  $\mathcal{G}$ 
6:   Remove all incident edges of each vertex in  $\mathcal{I}$  from  $\mathcal{G}'$ 
7:   Remove the set  $\mathcal{I}$  of vertices from  $\mathcal{G}'$ 
8:    $c = c + 1$ 
9: end while
    
```

²Recall that the dual variables can be generated only from linear programs, meaning that the integrality constraints (7) are replaced by constraints (11).

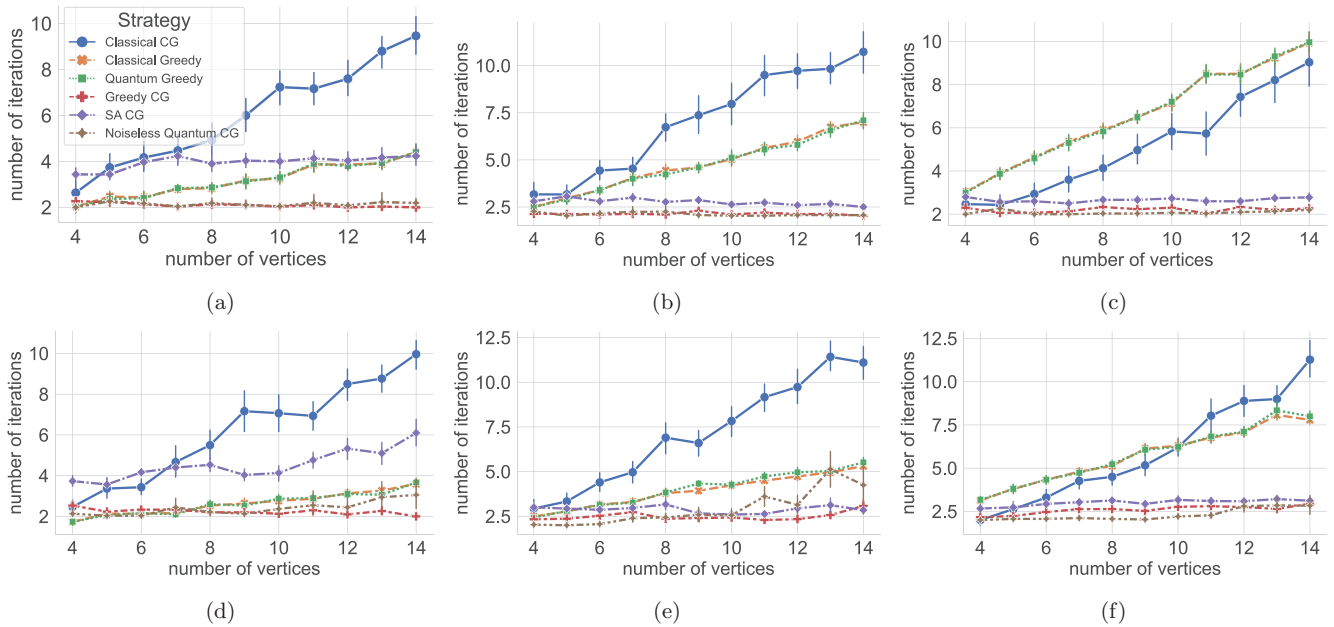


FIG. 7. Number of iterations before reaching the stop criteria on different graph classes (UD and non-UD), orders (from four up to 14 vertices), and densities (20%, 50%, and 80% of all possible connections) by applying different approaches: classical column generation (CG), greedy CG, SA CG, quantum CG, classical greedy, and quantum greedy. Results are shown on (a)–(c) unit-disk graphs and (d)–(f) non-UD graphs, as well as on (a) and (d) 20%-density graphs, (b) and (e) 50%-density graphs, and (c) and (f) 80%-density graphs.

since the size of each independent set gets smaller as the set of edges gets larger. Hence, in general, more iterations have to be done to cover all vertices of a dense graph. Also, while outperforming the classical CG approach on almost every graph class (in terms of the number of iterations), both classical and quantum greedy algorithms have their performance slightly decreased on UD graphs. Finally, taking advantage of the related superposition aspect, the proposed quantum CG outperforms all other approaches on all graph classes. For instance, while the quantum CG algorithm needs fewer than four (six) sampling iterations for all sparse and dense (0.5 density) non-UD graphs, quantum and classical greedy approaches (classical CG algorithm) need up to ten (12) pricing interactions to solve the same graph class.

Figure 8 shows the average gap³ (and the standard deviation with 95% of confidence interval) between the optimal solution and the best one found by applying the presented approaches. First, we observe that our proposed quantum CG approach has the best overall performance. Indeed, it could find the optimal solution in almost all instances; as seen in Fig. 8(f), our approach could not find the best solution only for some 13-vertex non-UD graphs. Also, unlike all other

approaches, the quantum CG is not impacted by the graph class; while the classical CG could better perform on dense graphs [see Figs. 8(c) and 8(f)], both classical and quantum greedy approaches are more stable on UD graphs. Also, as depicted in Figs. 8(d) and 8(a), the proposed quantum CG algorithm could reduce the average gap on 12-vertex non-UD (13-vertex UD) graphs from roughly 19% (11%) to 0% when compared to the quantum greedy (classical CG) approach.

Our proposed quantum pricing-based approach also outperforms both stochastic classical approaches in most of the instances, especially those related to UD and sparse graphs. Even though the quality of the solutions remains the same (see Fig. 8), the noiseless quantum CG could reduce by 50% the number of iterations on sparse graphs when compared to SA-based pricing, as observed in Figs. 7(a) and 7(d). Even though the greedy CG has fewer pricing iterations on some graph classes, as in bigger non-UD graphs with 20% and 50% of density [see Figs. 7(d) and 7(e)], the average gap could be reduced by 80% when our proposed quantum CG is applied on the same graph classes, as we observe in Figs. 8(a) and 8(d). This indicates that random sampling to find independent sets cannot solve pricing subproblems effectively.

C. Noisy and noiseless simulations

We now present the results from our numerical simulations wherein emulated noise is added as previously discussed. Figure 9 depicts the number of iterations before reaching the stop criteria and the gap between the final solution and the optimal one on different graph orders (from four up to 14 vertices) by applying different approaches: classical CG, greedy CG, SA CG, noiseless quantum CG, and noisy quantum CG. The results shown are related to applying a given

³Throughout the paper, the gap is calculated as follows: For a given problem instance and the value x_o of the optimal solution (i.e., the minimum number of colors needed to color the graph, also known as the chromatic number of the graph) and the value x_m of the solution (i.e., the number of colors used to color the graph), after running a given method the gap is given by $\frac{x_o - x_m}{x_o}$. In other words, the gap represents the distance between a given solution found by any method from the optimal one.

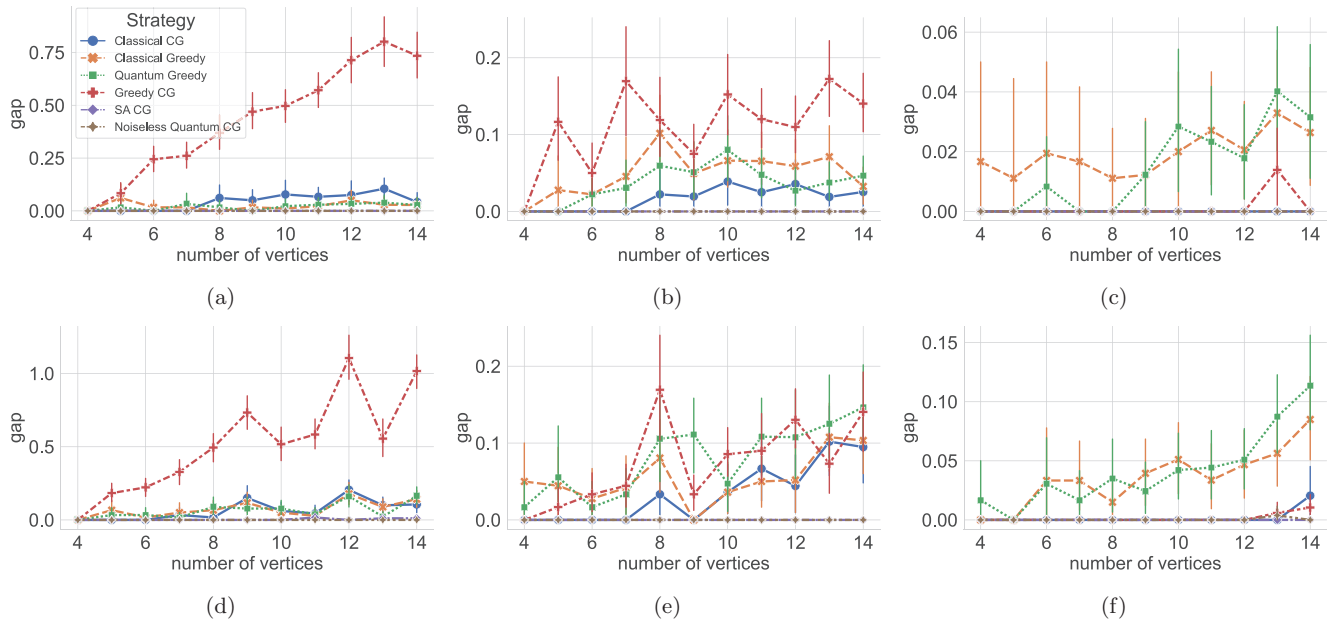


FIG. 8. Gap between the best solution found and the optimal one on different graph classes (UD and non-UD), orders (from four up to 14 vertices), and densities (20%, 50%, and 80% of all possible connections) by applying different approaches: classical column generation (CG), greedy CG, SA CG, noiseless quantum CG, classical greedy, and quantum greedy. Results are shown on (a)–(c) unit-disk graphs and (d)–(f) non-UD graphs, as well as on (a) and (d) 20%-density graphs, (b) and (e) 50%-density graphs, and (c) and (f) 80%-density graphs.

approach on all graph instances of the same order, regardless of their density and whether they are unit disks. First, as seen in Fig. 9(a), no impact is observed when noise is added to the quantum independent set sampler. Indeed, the overall final gap is similar to the noiseless model, also outperforming the classical CG. Finally, as observed in Fig. 9(b) the number of iterations needed to find the optimal solution of the relaxed RMP is increased by only 6% on average when the noisy model is compared to the noiseless one, even on big graphs.

VIII. CONCLUSION

In this study, we demonstrated that it is possible to incorporate quantum elements into classical state-of-the-art algorithms in order to improve their performances. Compared to the classical column generation, our neutral-atom-based quantum pricing could reduce by up to 83% the number of iterations needed to solve the minimum vertex Coloring problem. Also, our proposed hybrid approach could reduce the average gap from 19% to 0% when compared to both classical and greedy approaches. Moreover, unlike the deterministic approaches, our quantum pricing-based column generation is robust to all graph classes, including non-unit-disk graphs of all tested orders and sizes. This indicates that the proposed hybrid algorithm can efficiently solve other combinatorial problems, e.g., minimum edge coloring and minimum clustering problems, after some trivial translation-based preprocessing. The proposed framework also outperformed other stochastic algorithms, especially on sparse graphs. Indeed, our quantum pricing-based column generation could reduce by up to 50% the number of pricing iterations and by up to 80% the gap of the final solution on graphs when compared to the simulated-annealing-based

and greedy-based pricing approaches, respectively. Finally, we observed that our proposed hybrid framework is robust to noise. Indeed, the quality of the final solutions was not impacted when compared to the noiseless model, which is a great indication of how noise resilient the analog mode of operation can be. Our proposed approach can readily be implemented on neutral-atom quantum computing hardware.

Let us recall that our proposed algorithm remains a heuristic approach to solving combinatorial problems. Even though it can provide high-quality solutions to a plurality of instance classes, embedding this method into a branch-and-pricing framework is necessary to guarantee optimality to any instance of the problem under consideration. Since providing the initial subset of variables for the reduced master problem is an important step in any column-generation-based algorithm, the proposed quantum sampler can be used as a warm starter to generate such a subset of elements, e.g., independent sets. In the case of solving the minimal vertex coloring problem, by setting all vertex weights to 1, this approach might be useful in scenarios where QPU resources are limited. Also, this strategy might potentially speed up both classical and quantum column-generation approaches. Moreover, an optimal control-based strategy to design pulses to each input instance might potentially be applied to solve nontrivial pricing subproblems. Similarly, different register embedding approaches can be developed to take into consideration different information from the input data.

Note added. Recently, we became aware of a related work [52].

ACKNOWLEDGMENTS

We thank J. Bernos and V. Elfving for insightful discussions.

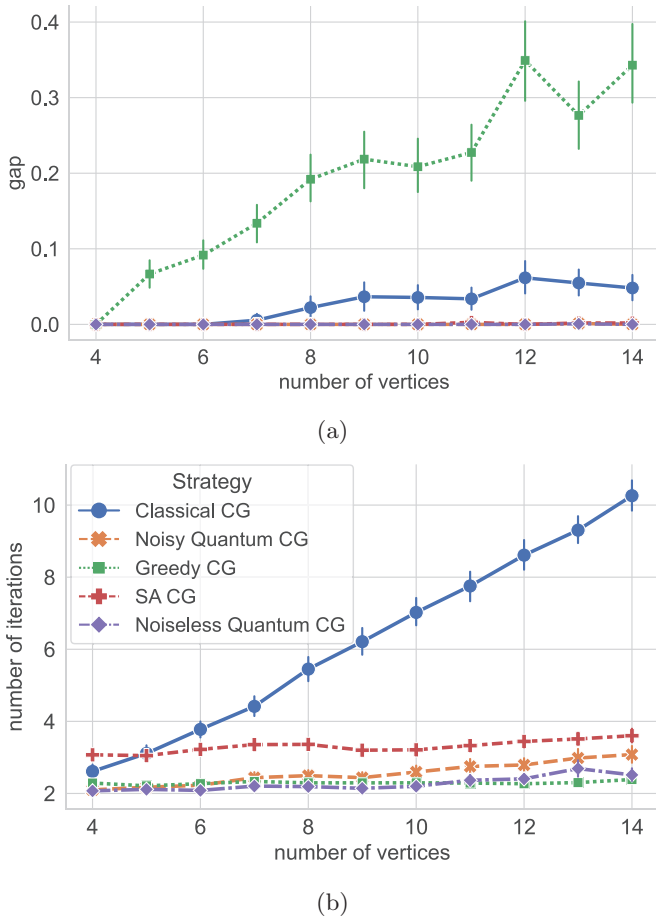


FIG. 9. Number of iterations before reaching the stop criteria and the gap between the final solution and the optimal one on different graph orders (from four up to 14 vertices) applying different approaches: classical column generation (CG), greedy CG, SA CG, noiseless quantum CG, and noisy quantum CG. (a) Gap between the final solution and the optimal one. (b) Number of iterations before reaching the stop criteria.

APPENDIX A: GREEDY ALGORITHM FOR THE MINIMUM VERTEX COLORING PROBLEM

We present here a greedy heuristic based on the minimum vertex coloring framework introduced in [35]. The main idea relies on interactively solving the maximum independent set problem with only a subset of the vertices of a given graph. Algorithm 2 summarizes the proposed approach.

As input, Algorithm 2 receives a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a set C of available colors. Note that, to always have a feasible color assignment, $|\mathcal{V}| \leq |C|$ must hold. First, a copy of \mathcal{G} is made with an auxiliary graph $\mathcal{G}' \equiv \mathcal{G}$. A variable k is also created; it keeps the index of the first available color. Then steps 3–7 are done until no vertex remains in \mathcal{G}' . In each iteration, a feasible independent set in \mathcal{G}' , potentially a maximal one, is generated. For instance, a classical solver can be used to solve the formulation (4) or one may use the proposed quantum sampler described in Sec. VIC by setting the algorithm to output only the largest sampled independent set. Then all vertices of \mathcal{G} with the same indices as those in the found independent set \mathcal{I} are colored with the first available

color from C , whose index is given by the variable c . Next all vertices of \mathcal{I} (and the related incident edges) are removed from \mathcal{G}' and then the reference of the first available color is updated (see step 8).

It is worth mentioning that, even though the number of qubits needed to run this algorithm on a QPU is reduced when compared to other approaches,⁴ its performance can be strongly impacted by the order in which the independent sets are generated. Also, in the worst case, $|\mathcal{V}|$ iterations can be done on the quantum device (take a complete graph as an example) before the final solution is found. More details about its performance are presented in Sec. VII.

APPENDIX B: FORCE-DIRECTED ALGORITHM

As pointed in [3], algorithms based on force-directed principles are normally used to embed graphs into planes in such a way that two connected (disjoint) vertices are placed close to (far from) each other, with a minimum (maximum) distance between them (from the plane’s center). In this context, in order to reflect inherent symmetries, Fruchterman and Reingold [54] also proposed an efficient algorithm to place the vertices evenly in the plane, making the edges’ lengths uniform. For this purpose, each edge from the graph is treated as a spring that holds its end-point vertices close to each other while a competing repulsive force is applied to push all vertices away from one another, even though they are not connected by an edge in the original graph. After enough iterations, the final system will reach equilibrium, minimizing then the difference between all attractive and repulsive forces.

The repulsive and attractive forces f_r and f_a between two vertices are given by Eqs. (B1) and (B2), respectively. While $k = \sqrt{A/|\mathcal{V}|}$ is set to be related to the area A of the Euclidean plane, d_{uv} is the distance between the vertices $u, v \in \mathcal{V}$. Finally, the total energy f_t of the system is given by adding the forces between all pairs of vertices, as shown in (B3). Therefore, f_t goes to zero as the system approaches its equilibrium. This register embedding is hereafter referred to as a spring layout

$$f_r(u, v) = -\frac{k^2}{r_{uv}}, \tag{B1}$$

$$f_a(u, v) = \frac{d_{uv}^2}{k}, \tag{B2}$$

$$f_t = \sum_{u, v \in \mathcal{E}} f_a(u, v) + \sum_{u \in \mathcal{V}} \sum_{v \in \mathcal{V}: u \neq v} f_r(u, v). \tag{B3}$$

Figure 5(a) depicts a possible embedding by directly applying the algorithm on a graph with five vertices and seven edges. For a deep description of the algorithm, one may refer to [3,54]. If such a realization exists, the graph under consideration is naturally embedded as a UD graph if enough iterations

⁴While some QUBO formulations for the vertex coloring problem present $|\mathcal{V}|^2$ variables, and hence at least $|\mathcal{V}|^2$ qubits are needed to encode their counterpart [27,53], e.g., when no auxiliary qubit is used, the greedy algorithm presented in [35] uses at most $|\mathcal{V}|$ variables or qubits.

Algorithm 3. Random independent set generator.

Input: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, W , a threshold weight w_{min} , and the maximum number of tries t

Output: A set of weighted independent sets

```

1:  $S \leftarrow \emptyset$ 
2: while the maximum number of tries  $t$  is not reached do
3:    $\mathcal{G}' \leftarrow \mathcal{G}$ 
4:    $\mathcal{I} \leftarrow \emptyset$ 
5:   while  $\mathcal{G}'$  has vertices do
6:     Randomly select a vertex in  $\mathcal{G}'$ 
7:     Add a selected vertex to  $\mathcal{I}$ 
8:     Remove the selected vertex and its neighbors from  $\mathcal{G}'$ 
9:   end while
10:  if  $\mathcal{I}$  was not generated before and its total weight
    is greater than  $w_{min}$  then
11:     $S \leftarrow S \cup \mathcal{I}$ 
12:  end if
13: end while
14: return  $S$ 

```

are allowed, i.e., by iterating until the system reaches equilibrium. It is also worth mentioning that, as pointed out in [3],

the proposed embedding strategy is only feasible on neutral-atom-based QPUs once they respect the device's technical constraints, such as maximum distance from the register's center and minimum distance between atoms. If any technical constraint is violated, one might rescale every position vector by a factor $\alpha > 0$.

APPENDIX C: CLASSICAL GREEDY PRICING ALGORITHM

Algorithm 3 summarizes the proposed random weighted independent set generator. As input, the algorithm receives a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, where W is the vertex weighting vector. First, an auxiliary graph \mathcal{G}' is created to receive a copy of the input graph \mathcal{G} (step 3). Then a vertex from graph \mathcal{G}' is randomly selected and added to the independent set S (see steps 6 and 7). Next the selected vertex and its neighbors are removed from \mathcal{G}' in step 8. Steps 6–8 are repeated until no vertex remains. The overall cost of the final solution is then calculated considering the vertex weighting vector W (see step 11). Due to the inherently stochastic nature of the proposed algorithm, different weighted independent sets might potentially be generated from the same input by running steps 3–11 multiple times (see the condition in step 2).

-
- [1] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, Branch-and-price: Column generation for solving huge integer programs, *Oper. Res.* **46**, 316 (1998).
 - [2] A. Bachem and W. Kern, *Linear Programming Duality* (Springer, Berlin, 1992), pp. 89–111.
 - [3] W. da Silva Coelho, M. D'Arcangelo, and L.-P. Henry, Efficient protocol for solving combinatorial graph problems on neutral-atom quantum processors, [arXiv:2207.13030](https://arxiv.org/abs/2207.13030).
 - [4] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, Superconducting qubits: Current state of play, *Annu. Rev. Condens. Matter Phys.* **11**, 369 (2020).
 - [5] M. H. Devoret, A. Wallraff, and J. M. Martinis, Superconducting qubits: A short review, [arXiv:cond-mat/0411174](https://arxiv.org/abs/cond-mat/0411174).
 - [6] I. Pogorelov, T. Feldker, C. D. Marciniak, L. Postler, G. Jacob, O. Kriegelsteiner, V. Podlesnic, M. Meth, V. Negnevitsky, M. Stadler, B. Höfer, C. Wächter, K. Lakhmanskiy, R. Blatt, P. Schindler, and T. Monz, Compact ion-trap quantum computing demonstrator, *PRX Quantum* **2**, 020343 (2021).
 - [7] J. Benhelm, G. Kirchmair, C. F. Roos, and R. Blatt, Towards fault-tolerant quantum computing with trapped ions, *Nat. Phys.* **4**, 463 (2008).
 - [8] C. Antón, J. C. Loredó, G. Coppola, H. Ollivier, N. Viggianiello, A. Harouri, N. Somaschi, A. Crespi, I. Sagnes, A. Lemaître, L. Lanco, R. Osellame, F. Sciarrino, and P. Senellart, Interfacing scalable photonic platforms: Solid-state based multi-photon interference in a reconfigurable glass chip, *Optica* **6**, 1471 (2019).
 - [9] L. Henriët, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak, Quantum computing with neutral atoms, *Quantum* **4**, 327 (2020).
 - [10] J. Wurtz, P. Lopes, N. Gemelke, A. Keesling, and S. Wang, Industry applications of neutral-atom quantum computing solving independent set problems, [arXiv:2205.08500](https://arxiv.org/abs/2205.08500).
 - [11] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications* (Macmillan, London, 1976), Vol. 290.
 - [12] M. Roth, A. Ben-David, D. Deutscher, G. Flysher, I. Horn, A. Leichtberg, N. Leiser, Y. Matias, and R. Merom, *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, 2010), pp. 233–242.
 - [13] A. Kershenbaum, *Telecommunications Network Design Algorithms* (McGraw-Hill, New York, 1993).
 - [14] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout design, *Oper. Res.* **36**, 493 (1988).
 - [15] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, Noisy intermediate-scale quantum algorithms, *Rev. Mod. Phys.* **94**, 015004 (2022).
 - [16] H. Nishi, T. Kosugi, and Y.-i. Matsushita, Implementation of quantum imaginary-time evolution method on NISQ devices by introducing nonlocal approximation, *npj Quantum Inf.* **7**, 85 (2021).
 - [17] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, X.-Z. Luo, B. Nash, X. Gao, B. Barak, E. Farhi, S. Sachdev, N. Gemelke, L. Zhou, S. Choi, H. Pichler *et al.* Quantum optimization of maximum independent set using Rydberg atom arrays, *Science* **376**, 1209 (2022).
 - [18] M. Kim, K. Kim, J. Hwang, E.-G. Moon, and J. Ahn, Rydberg quantum wires for maximum independent set problems with nonplanar and high-degree graphs, *Nat. Phys.* **18**, 755 (2022).

- [19] B. Albrecht, C. Dalyac, L. Leclerc, L. Ortiz-Gutiérrez, S. Thabet, M. D'Arcangelo, V. E. Elfving, L. Lassablière, H. Silvério, B. Ximenez, L.-P. Henry, A. Signoles, and L. Henriët, Quantum feature maps for graph machine learning on a neutral atom quantum processor, [arXiv:2211.16337](https://arxiv.org/abs/2211.16337).
- [20] L. Novo, S. Chakraborty, M. Mohseni, and Y. Omar, Environment-assisted analog quantum search, *Phys. Rev. A* **98**, 022316 (2018).
- [21] C. Dalyac, L. Henriët, E. Jeandel, W. Lechner, S. Perdrix, M. Porcheron, and M. Veshchezerova, Qualifying quantum approaches for hard industrial optimization problems. A case study in the field of smart-charging of electric vehicles, *EPJ Quantum Technol.* **8**, 12 (2021).
- [22] A. Rajak, S. Suzuki, A. Dutta, and B. K. Chakrabarti, Quantum annealing: An overview, *Philos. Trans. R. Soc. A* **381**, 20210417 (2022).
- [23] A. Cornejo and F. Kuhn, in *International Symposium on Distributed Computing*, edited by N. A. Lynch and A. A. Shvartsman, Lecture Notes in Computer Science Vol. 6343 (Springer, Berlin, 2010), pp. 148–162.
- [24] D. Marx, Graph colouring problems and their applications in scheduling, *Period. Polytech. Electr. Eng. (Archives)* **48**, 11 (2004).
- [25] R. M. Karp, in *Complexity of Computer Computations*, edited by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger (Springer, Boston, 1972), pp. 85–103.
- [26] K. Kudo, Constrained quantum annealing of graph coloring, *Phys. Rev. A* **98**, 022301 (2018).
- [27] S. M. Ardelean and M. Udrescu, Graph coloring using the reduced quantum genetic algorithm, *PeerJ Comput. Sci.* **7**, e836 (2022).
- [28] O. Titiloye and A. Crispin, Quantum annealing of the graph coloring problem, *Discrete Optim.* **8**, 376 (2011).
- [29] C. Silva, A. Aguiar, P. Lima, and I. Dutra, Mapping graph coloring to quantum annealing, *Quantum Mach. Intell.* **2**, 16 (2020).
- [30] Z. Tabi, K. H. El-Safy, Z. Kallus, P. Hága, T. Kozsik, A. Glos, and Z. Zimborás, *Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, Piscataway, 2020), pp. 56–62.
- [31] J. Kwok and K. Pudenz, Graph coloring with quantum annealing, [arXiv:2012.04470](https://arxiv.org/abs/2012.04470).
- [32] A. Fabrikant and T. Hogg, in *Proceedings of the 18th National Conference on Artificial Intelligence, Edmonton, 2002*, edited by R. Dechter, M. Kearns, and R. Sutton (AAAI, Menlo Park, 2002), pp. 22–27.
- [33] K. Shimizu and R. Mori, Exponential-time quantum algorithms for graph coloring problems, *Algorithmica* **84**, 3603 (2022).
- [34] A. Ambainis, K. Balodis, J. Iraids, M. Kokainis, K. Prūsis, and J. Vihrovs, *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms* (SIAM, Philadelphia, 2019), pp. 1783–1793.
- [35] G. Vitali, P. Viviani, C. Vercellino, A. Scarabosio, A. Scionti, O. Terzo, E. Giusto, and B. Montrucchio, The International Conference for High Performance Computing, Networking, Storage, and Analysis, Research Posters (unpublished), available at <https://sc21.supercomputing.org/presentation/?id=rpost113&sess=sess278>.
- [36] J. Ossorio-Castillo and F. Pena-Brage, Optimization of a refinery scheduling process with column generation and a quantum annealer, *Optim. Eng.* **23**, 1471 (2022).
- [37] H. Pichler, S.-T. Wang, L. Zhou, S. Choi, and M. D. Lukin, Quantum optimization for maximum independent set using Rydberg atom arrays, [arXiv:1808.10816](https://arxiv.org/abs/1808.10816).
- [38] S. Martiel, T. Ayrat, and C. Allouche, Benchmarking quantum coprocessors in an application-centric, hardware-agnostic, and scalable way, *IEEE Trans. Quantum Eng.* **2**, 1 (2021).
- [39] M. Balinski and A. W. Tucker, Duality theory of linear programs: A constructive approach with applications, *SIAM Rev.* **11**, 347 (1969).
- [40] H. Breu and D. G. Kirkpatrick, Unit disk graph recognition is NP-hard, *Comput. Geom.* **9**, 3 (1998).
- [41] Using QAOA and QAA to solve a UD-MIS problem (unpublished).
- [42] T. Albash and D. A. Lidar, Adiabatic quantum computation, *Rev. Mod. Phys.* **90**, 015002 (2018).
- [43] D. Bluvstein, A. Omran, H. Levine, A. Keesling, G. Semeghini, S. Ebadi, T. T. Wang, A. A. Michailidis, N. Maskara, W. W. Ho, S. Choi, M. Serbyn, M. Greiner, V. Vuletić, and M. D. Lukin, Controlling quantum many-body dynamics in driven Rydberg atom arrays, *Science* **371**, 1355 (2021).
- [44] V. Batagelj and U. Brandes, Efficient generation of large random networks, *Phys. Rev. E* **71**, 036113 (2005).
- [45] M. Penrose, *Random Geometric Graphs* (Oxford University Press, Oxford, 2003), Vol. 5.
- [46] E. Malaguti and P. Toth, A survey on vertex coloring problems, *Int. Trans. Oper. Res.* **17**, 1 (2010).
- [47] GLPK (GNU Linear Programming Kit, <https://www.gnu.org/software/glpk/>), accessed 5 November 2022.
- [48] D-Wave Simulated Annealing QUBO Sampler, <https://docs.ocean.dwavesys.com/projects/neal/en/latest/reference/sampler.html>, accessed 21 December 2022.
- [49] A. Hagberg, P. Swart, and D. S. Chult, Exploring network structure, dynamics, and function using NetworkX, Los Alamos National Laboratory Report No. LA-UR-08-05495, 2008 (unpublished).
- [50] H. Silvério, S. Grijalva, C. Dalyac, L. Leclerc, P. J. Karalekas, N. Shammah, M. Beji, L.-P. Henry, and L. Henriët, Pulser: An open-source package for the design of pulse sequences in programmable neutral-atom arrays, *Quantum* **6**, 629 (2022).
- [51] L. Leclerc, L. Ortiz-Gutiérrez, S. Grijalva, B. Albrecht, J. R. K. Cline, V. E. Elfving, A. Signoles, L. Henriët, G. Del Bimbo, U. A. Sheikh, M. Shah, L. Andrea, F. Ishtiaq, A. Duarte, S. Mugel, I. Caceres, M. Kurek, R. Orus, A. Seddik, O. Hammami, H. Isselane, and D. M'tamon, Financial risk management on a neutral atom quantum processor, [arXiv:2212.03223](https://arxiv.org/abs/2212.03223).
- [52] M. Veshchezerova, Quantum algorithms for energy management optimization problems, Ph.D. dissertation, Ecole Doctorale IAEM, 2022.
- [53] F. Glover, G. Kochenberger, R. Hennig, and Y. Du, Quantum bridge analytics I: A tutorial on formulating and using QUBO models, *Ann. Oper. Res.* **314**, 141 (2022).
- [54] T. M. Fruchterman and E. M. Reingold, Graph drawing by force-directed placement, *Softw. Pract. Exper.* **21**, 1129 (1991).