Quantum Optimization with Arbitrary Connectivity Using Rydberg Atom Arrays

Minh-Thi Nguyen,[1,‡] Jin-Guo Liu,[2,1,‡] Jonathan Wurtz,[1] Mikhail D. Lukin,[2] Sheng-Tao Wang,[1,*] and Hannes Pichler[3,4,†]

[1] *QuEra Computing Inc., 1284 Soldiers Field Road, Boston, Massachusetts 02135, USA*
[2] *Department of Physics, Harvard University, Cambridge, Massachusetts 02138, USA*
[3] *Institute for Theoretical Physics, University of Innsbruck, Innsbruck A-6020, Austria*
[4] *Institute for Quantum Optics and Quantum Information, Austrian Academy of Sciences, Innsbruck A-6020, Austria*

(Received 14 September 2022; revised 14 December 2022; accepted 4 January 2023; published 14 February 2023)

Programmable quantum systems based on Rydberg atom arrays have recently been used for hardware-efficient tests of quantum optimization algorithms [Ebadi *et al.*, Science, **376**, 1209 (2022)] with hundreds of qubits. In particular, the maximum independent set problem on so-called unit-disk graphs, was shown to be efficiently encodable in such a quantum system. Here, we extend the classes of problems that can be efficiently encoded in Rydberg arrays by constructing explicit mappings from a wide class of problems to maximum-weighted independent set problems on unit-disk graphs, with at most a quadratic overhead in the number of qubits. We analyze several examples, including maximum-weighted independent set on graphs with arbitrary connectivity, quadratic unconstrained binary optimization problems with arbitrary or restricted connectivity, and integer factorization. Numerical simulations on small system sizes indicate that the adiabatic time scale for solving the mapped problems is strongly correlated with that of the original problems. Our work provides a blueprint for using Rydberg atom arrays to solve a wide range of combinatorial optimization problems with arbitrary connectivity, beyond the restrictions imposed by the hardware geometry.

DOI: 10.1103/PRXQuantum.4.010316

I. INTRODUCTION

Quantum optimization algorithms aim to solve combinatorial optimization problems [1,2] by utilizing controlled dynamics of quantum many-body systems. The key idea underlying this paradigm is to steer the dynamics of quantum systems such that their final states provide solutions to the optimization problem of interest. Such dynamics are often achieved either via the adiabatic principle in quantum annealing algorithms (QAAs) [3–7], or by employing more general, variational approaches, as exemplified by quantum approximate optimization algorithms (QAOAs) [8]. A popular approach to design such quantum algorithms is to formulate the optimization problem in terms of a classical spin model [9] that can be implemented on special-purpose quantum hardware.

An exciting possibility in this context is offered by Rydberg atom arrays [10]. Owing to the Rydberg-blockade mechanism [10–12], these systems realize spin models that naturally encode a paradigmatic combinatorial optimization problem, namely the maximum independent set (MIS) problem on a special class of geometric graphs, called unit-disk graphs (UDGs) [13]. This allows a direct implementation of a variety of quantum optimization algorithms on this platform [10,13,14]. Remarkably, first experiments exploring this approach [10] observed a superlinear quantum speedup over optimized classical simulated annealing for finding exact solutions for some of the hardest accessible graphs. However, the restriction to unit-disk graphs limits the applicability of this approach. Overcoming this limitation is one of the major challenges for exploring quantum optimization on a much wider range of optimization problems, including several problems of industrial relevance [15].

Approaches to extend the applicability of Rydberg atom arrays beyond UDGs have been recently explored in Refs. [14,16], but they are either limited to a specific class of graphs [16], or require three-dimensional arrays [14],

*swang@quera.com
†hannes.pichler@uibk.ac.at
‡These authors contributed equally to this work.

and both require bespoke encoding for each problem graph with unclear overhead for general graphs. Alternatively, other schemes that can map arbitrary nonlocal interactions into local ones have been proposed [17,18], but their implementations are experimentally even more demanding, requiring either four-body interactions [17,19] or the use of tunable Ising interactions [18].

In this paper, we introduce a new, systematic approach to encode optimization problems with arbitrary connectivity into Rydberg atom arrays. Our scheme requires only two-dimensional (2D) atom trapping and the Rydberg-blockade mechanism as the main ingredients, both of which have been demonstrated already on current Rydberg atom array platforms with high fidelity [10] (see Fig. 1). Importantly, our encoding is constructive and efficient as it incurs only a minimal, quadratic overhead in the number of qubits. We specifically discuss our approach on the paradigmatic optimization problems including maximum-weight independent set (MWIS) problems on arbitrary graphs, arbitrary quadratic unconstrained binary optimization (QUBO) or Ising problems. In addition, we apply our method to generic constraint-satisfaction problems and show how integer factorization can be mapped to Rydberg atom arrangements. Finally, we perform numerical simulations on small system sizes comparing the adiabatic time scale for original MWIS on non-UDG graphs to that of the mapped problem and observe strong correlations that suggest the encoding does not negatively impact the performance of quantum algorithms.

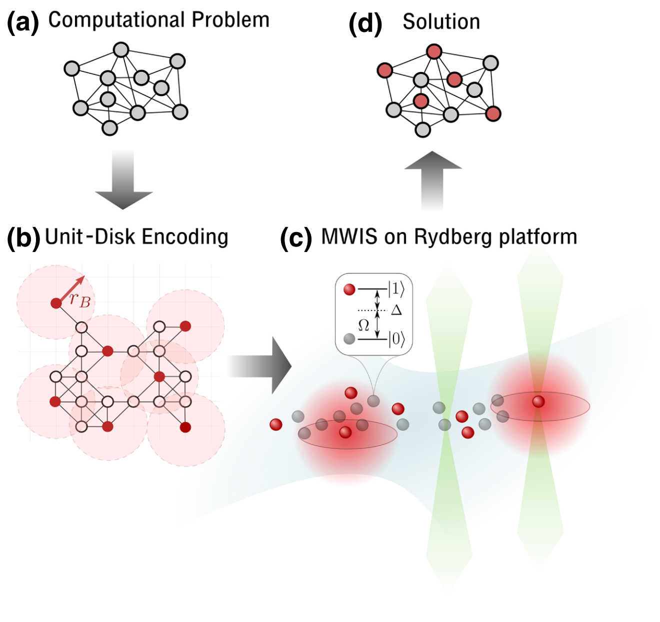


FIG. 1. Procedure to solve a variety of optimization problems using programmable Rydberg atom arrays. The original computational problem (a) can be mapped onto a MWIS problem on a UDG in (b). (c) Physical platform, where each vertex in (b) represents an atom trapped by optical tweezers. Each two-level atom can be coherently driven with Rabi frequency Ω and detuning Δ, and the Rydberg-blockade mechanism prevents two atoms from being simultaneously excited to state $|1\rangle$ if they are within a unit distance r_B. (d) The solution to the UDG-MWIS problem encodes the solution to the original problem.

We note that MIS on UDGs is known to be NP-complete [20], so in principle any NP problem can be reduced to MIS on UDGs with a polynomial overhead. Specific, formal reduction sequences have, for example, been considered in Ref. [10], but direct application of the prescribed reduction method requires at least $O(N^6)$ overhead. It is important for near-term implementation on quantum machines to find a low-overhead, explicit mapping, which is the main result of our work.

II. OVERVIEW OF MAIN RESULTS

In this section, we provide an overview of the main results of this work. The main ideas are summarized in Figs. 1 and 2. Given a computation problem, we map it to a UDG-MWIS problem (i.e., a MWIS problem on a UDG) using a novel encoding scheme. The resulting UDG-MWIS is the native problem for Rydberg atom array platforms allowing a direct implementation of QAA or QAOA for its solution [10,13]. The solution for the UDG-MWIS obtained on the quantum device can then be mapped back to a solution for the original computation problem. The key result of this work is to provide a general framework for the low-overhead, efficient, and explicit mapping.

The main idea underlying our encoding is summarized in Fig. 2 for three examples discussed in detail in this work: MWIS on general (non-unit-disk) graphs, QUBO and Ising problems with arbitrary connectivity, and integer factorization [21]. The general framework for mapping combinatorial optimization problems defined on graphs can be seen in Figs. 2(a)–2(d). First, the variables corresponding to vertices in the original graph can be encoded in one-dimensional chains of atoms using the *copy gadget.* These chains (represented by lines) are then arranged in the form of a crossing lattice shown in Fig. 2(b), exhibiting exactly one crossing between each pair of lines. For each such crossing, we use additional gadgets—the *crossing gadget* and the *crossing-with-edge gadget*— to encode the presence (and strength) or absence of an interaction for each pair of lines. All these gadgets are carefully designed such that the resulting graph is a UDG (embedded on a square lattice), and the solution of the original problem is encoded in its MWIS. The mapping for the factoring problem follows a similar strategy [Figs. 2(e)–2(g)]: we first encode the problem of finding the prime factors of a N-bit integer into an optimization problem; with the help of a crossing lattice and a properly designed factoring gadget, this optimization problem is then transformed to a UDG-MWIS problem. In all cases, the overhead in the number of qubits is at most $O(N^2)$, which is optimal for arbitrary connectivity [22].

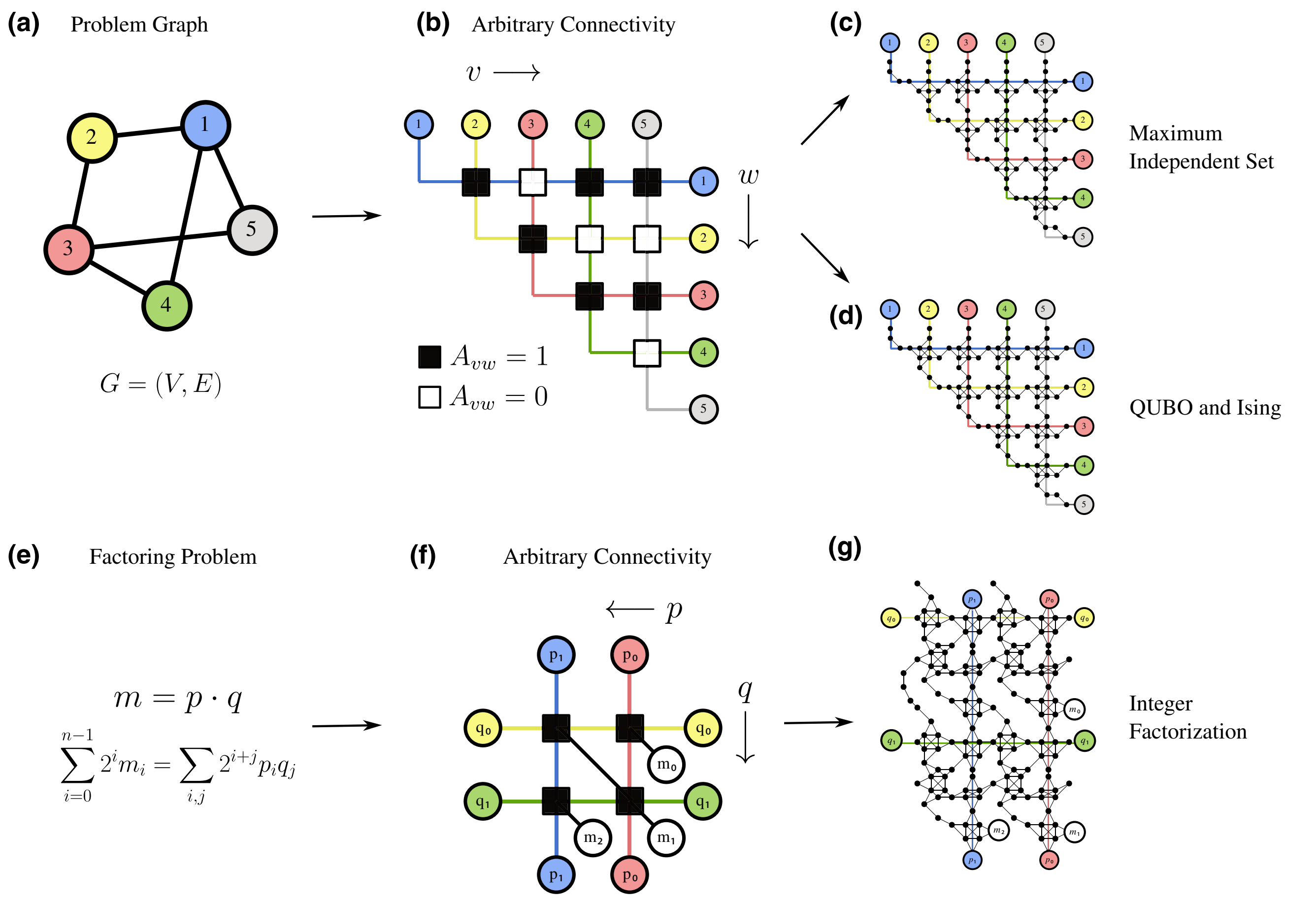


FIG. 2. Architecture of UDG-MWIS mapping for three example problems. (a) Example non-UDG represented by $G = (V, E)$ for the MWIS and QUBO problems. (b) Crossing lattice used to construct UDG-MWIS mappings. Vertices in (a) are binary variables that can be represented effectively by lines to construct the lattice. Intersections in the lattice allow arbitrary connectivity between the variables, abstractly represented by squares. The lattice mimics an upper triangular adjacency matrix A, where for two vertices $\{v, w\} \in V$, $A_{vw} = 1$ if $(v, w) \in E$ and $A_{vw} = 0$ otherwise, represented abstractly by a filled and empty square here, respectively. (c) Final UDG-MWIS representation of the original MWIS problem on general graphs. (d) Final UDG-MWIS representation of the original QUBO and Ising problem. (e),(f),(g) Similar encoding procedure for the integer factorization problem for the example $6 = 2 \times 3$, with the corresponding UDG-MWIS representation shown in (g).

The paper is organized as follows. Section III introduces Rydberg atom arrays and explains the natural encoding of UDG-MWIS on the platform. Section IV outlines the basic encoding gadgets by first showing MWIS gadgets for simple constraint-satisfaction problems. Then, some useful mapping gadgets are presented, including the *copy gadget*, the *crossing gadget*, and the *crossing-with-edge gadget*. Section V uses the above gadgets and the idea of a crossing lattice to construct the explicit mapping to UDG-MWIS from three example applications: MWIS on general graphs, QUBO, and integer factorization. Additional gadgets and problem-specific details are presented. Section VI studies the performance of quantum algorithms before and after the mapping, focusing on the example of the MWIS problem from a general graph to UDG mapping. Finally, Sec. VII concludes the paper and outlines some next steps and challenges. The Appendix includes more details on strategies for overhead reduction and simplification in Appendix A, a discussion on local defects and MWIS guarantees in Appendix B, a more efficient mapping for problems with local connectivity in Appendix C, and more details on the construction of the factoring gadget in Appendix D.

III. BACKGROUND

A. Rydberg atom arrays

This work is primarily motivated by recent advances in experiments with Rydberg atom arrays using neutral atoms in optical tweezers [16,23–33]. In these systems, atoms can be deterministically placed at programmable positions in two dimensions [10,26,27]. Each atom realizes a qubit with an internal ground state representing $|0\rangle$

and a highly excited, long-lived Rydberg state representing $|1\rangle$. The atoms can be coherently manipulated with laser fields and interact pairwise via induced dipole-dipole interactions when two atoms are in the Rydberg state. Specifically, the laser-induced quantum dynamics of this system can be described by the Hamiltonian

$$H_{\text{Ryd}} = \sum_{v} \frac{\Omega_v}{2}\sigma_v^x - \sum_{v} \Delta_v n_v + \sum_{v<w} V_{\text{Ryd}}(|\overrightarrow{\mathbf{r}_v} - \overrightarrow{\mathbf{r}_w}|) n_v n_w. \tag{1}$$

Here, $\overrightarrow{\mathbf{r}_v}$ denotes the position of the atom labeled by v, $\sigma_v^x = |1\rangle_v\langle 0| + |0\rangle_v\langle 1|$ coherently flips its internal state, and $n_v = |1\rangle_v\langle 1|$ counts if the atom is in the Rydberg state. The parameters Ω_v and Δ_v are the Rabi frequency and laser detuning for the vth atom. In experiments, the laser detuning can be controlled in a site-dependent way, for example, using local ac-Stark shifts [34]. The interaction potential $V_{\text{Ryd}}(|\overrightarrow{\mathbf{r}_v} - \overrightarrow{\mathbf{r}_w}|) = C_6/|\overrightarrow{\mathbf{r}_v} - \overrightarrow{\mathbf{r}_w}|^6$ leads to a strong (distance-dependent) energy penalty for configurations where two nearby atoms are simultaneously in the Rydberg state, giving rise to the so-called *Rydberg-blockade* mechanism [10–13]. As a result, the low-energy states of the Hamiltonian do not contain states with pairs of atoms that are both in the Rydberg state if they are within some characteristic distance, defined as the blockade radius r_B. This effect naturally imposes the independent set constraint on the ground state(s) of the Hamiltonian at $\Omega_v = 0$, which, as discussed below, allows one to encode the MWIS of a corresponding UDG [10,13]. In this classical limit ($\Omega_v = 0$), it is convenient to define the blockade radius r_B via $V_{\text{Ryd}}(r_B) = \max_v(\Delta_v)$, which is the convention we adopt throughout this work.

B. Unit-disk graphs

A unit-disk graph is a graph $G = (V, E)$ with vertices V and edges E that can be embedded in the 2D Euclidean plane such that two vertices are connected by an edge if and only if they are separated by a distance smaller than a unit radius. We are interested in unit-disk graphs since they are in one-to-one correspondence with atom arrangements in 2D. Specifically, each atom represents a vertex, and we identify the blockade radius with the unit-disk radius of the graph. In this way, the low-energy configurations of the atom array at $\Omega_v = 0$ correspond to large independent sets of the unit-disk graph [13].

C. Maximum-weight independent sets

An independent set of a graph G is the subset of vertices $S \subseteq V$, such that none of the vertices in S are connected by an edge in G. The largest such independent set is called a maximum independent set. Note that in general the MIS may not be unique. The problem of finding a MIS is called the maximum independent set problem. The MIS problem can be generalized to the maximum-weight independent set problem, where each vertex is assigned a weight $\delta_v > 0$, and accordingly, a weight W_S is assigned to each subset of vertices $S \subseteq V$ via $W_S = \sum_{v\in S}\delta_v$. The MWIS problem is to find an independent set with the largest weight. It can be formulated as an energy minimization problem. For this, one can associate a binary variable $n_v \in \{0, 1\}$ with each vertex $v \in V$. This allows us to identify a subset of vertices S by a bitstring $\mathbf{n} = (n_1, n_2, \ldots)$, via $S = \{v \in V | n_v = 1\}$. We frequently use this one-to-one correspondence between bitstings and subsets in the remainder of the paper. Using this representation, we can consider the cost function

$$H_{\text{MWIS}} = -\sum_{v\in V} \delta_v n_v + \sum_{(u,v)\in E} U_{uv} n_u n_v. \tag{2}$$

If $U_{uv} > \delta_w > 0$ for all u, v, w, the ground-state configuration of H_{MWIS} indeed corresponds to the MWIS. As in the unweighted case, the ground state can be degenerate, corresponding to multiple independent sets achieving the same maximum weight [35].

If the graph G is a UDG, we then refer to the corresponding MWIS problem as the UDG-MWIS problem. Importantly, for such UDG-MWIS problems, the Hamiltonian (2) coincides with the Rydberg atom array Hamiltonian (1) at $\Omega = 0$, where each atom is placed at the respective location of the corresponding vertex, the blockade radius is identified with the unit disk radius (and the interaction tails beyond the blockade radius is neglected [10,36]), and the weight of each vertex is identified with the local detuning $\Delta_v = \delta_v$ [10,13]. Hence, we aim in the following to encode a variety of problems of interests in UDG-MWIS.

IV. ENCODING GADGETS

In this section, we construct a number of encoding gadgets, which are the basic tools used in this work to reformulate a variety of optimization problems as UDG-MWIS. To this end, we first encode solutions of a set of elementary constraint-satisfaction problems as the solutions of a MWIS problem on properly constructed unit-disk graphs.

A. Constraint satisfaction problems as MWIS

Consider a set of binary variables $\mathbf{n} = (n_1, n_2, \ldots)$ with $n_i \in \{0, 1\}$ and a set of constraints between them, denoted by C, that can be simultaneously satisfied by one or more assignments. We represent this constraint-satisfaction problem as a MWIS problem by constructing a weighted graph G_C, such that the solutions are in correspondence with the maximum-weighted independent sets of G_C. More specifically, we say the MWIS problem on G_C represents the constraint-satisfaction problem C if every

MWIS of G_C coincides with a satisfying assignment of C, and if every satisfying assignment of C corresponds to at least one MWIS of G_C. Note that the number of vertices in G_C can be larger than the number of variables in C, in which case we require the correspondence between the MWISs and the satisfying assignments only on the subset of vertices that correspond to the variables in C. Below, we illustrate this concept on several examples.

1. Single constraints

We start with the simple example of two bits $\mathbf{n} = (n_1, n_2)$ with a single constraint

$$n_1 = \overline{n_2}, \tag{3}$$

where $\overline{n_i} \equiv 1 - n_i$ denotes the negation of n_i. This simple NOT constraint has two satisfying assignments, $\mathbf{n} = (1, 0)$ and $\mathbf{n} = (0, 1)$.

We represent this constraint-satisfaction problem as a MWIS problem on a graph with two equally weighted vertices connected by an edge [Fig. 3(a)], whose cost function is simply $H_{\text{MWIS}} = -\delta(n_1 + n_2) + U n_1 n_2$, with $U > \delta > 0$. This graph has two degenerate MWISs, $\mathbf{n} = (1, 0)$ and $\mathbf{n} = (0, 1)$, which are indeed in one-to-one correspondence to the two satisfying assignments of the constraint.

Note, that for these two assignments, the cost function evaluates to $H_{\text{MWIS}} = -\delta$, while a violation of the constraint incurs a cost $\delta > 0$ (for $\mathbf{n} = (0, 0)$) or a cost of $U - \delta > 0$ (for $\mathbf{n} = (1, 1)$), rendering them energetically unfavorable. For the remainder of the paper, we introduce the quantity $\delta_{\text{gap}} = \min(U - \delta, \delta) > 0$ as the minimum energy penalty for violation of a constraint.

Next, we consider another simple two-variable constraint:

$$n_1 n_2 = 0. \tag{4}$$

This constraint has three satisfying assignments $\mathbf{n} \in \{(0, 0), (1, 0), (0, 1)\}$ and may be represented as a MWIS problem on a complete graph with three vertices with equal weights [Fig. 3(b)]. The first two vertices, labeled 1 and 2, correspond to the two bits of interest, while the third vertex corresponds to an ancillary variable. The cost function associated with this MWIS problem is $H_{\text{MWIS}} = -\delta(n_1 + n_2 + n_3) + U(n_1 n_2 + n_2 n_3 + n_3 n_1)$, with the three degenerate solutions corresponding to the three satisfying assignments. Importantly, each of the three MWIS states coincides with one of the three satisfying assignments on the two vertices of interest [see Fig. 3(b)]. Again, a violation of the constraint incurs an energy cost of at least δ_{gap}.

We remark that in this manner one can construct the MWIS representation of all the basic operations in Boolean logic, by providing a gadget that is the MWIS representation of the NOR constraint in Fig. 3(c).

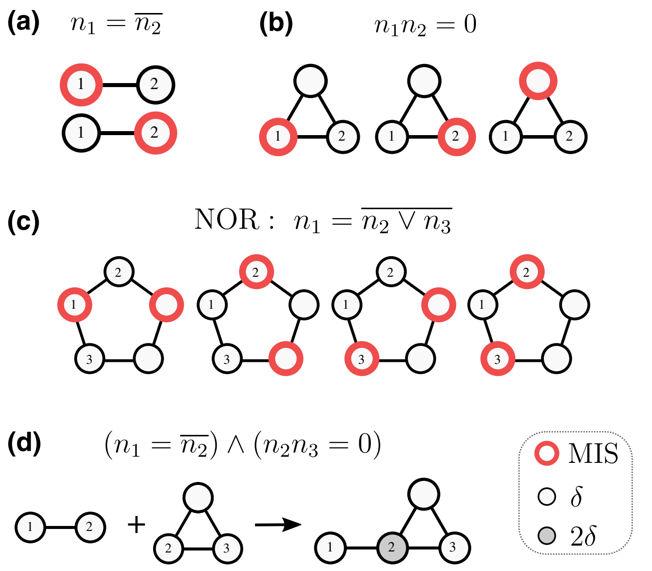

(a) $n_1 = \overline{n_2}$
(b) $n_1 n_2 = 0$
(c) NOR : $n_1 = \overline{n_2 \vee n_3}$
(d) $(n_1 = \overline{n_2}) \wedge (n_2 n_3 = 0)$
1
2
3
MIS
δ
2δ

FIG. 3. MWIS representation of some example constraints. Each bit is represented by a corresponding vertex in the MWIS problem graph. The weight of the vertices is indicated by its interior color on a gray scale. For each example, the degenerate MWIS configurations are shown by identifying vertices in a MWIS with a red boundary. The MWISs correspond to the satisfying assignments to the corresponding constraint-satisfaction problem. (a) MWIS representation of $n_1 = \overline{n_2}$. (b) MWIS representation of $n_1 n_2 = 0$, with the third, unlabeled vertex being an ancillary vertex. For each of the three MWIS states, the configuration on the relevant vertices 1 and 2 matches the corresponding satisfying assignment. (c) MWIS representation of the NOR constraint using two ancilla vertices. Note that only four of the five MWIS states are shown. Nevertheless, all five MWIS states correspond to the four satisfying assignments on the relevant vertices 1, 2, and 3. (d) A set of constraints $C = \{n_1 = \overline{n_2}, n_2 n_3 = 0\}$, where each is of the form given in (a),(b). The corresponding MWIS problem is obtained by combining the two graphs corresponding to both constraints in C, resulting in the weighted graph on the right. Observe that vertex 2, which appears in both constraints, has twice the weight of the other vertices.

2. Conjunction of constraints

Consider now a situation where C consists of a set of multiple constraints that have to be satisfied simultaneously. For example, consider a conjunction of constraints involving three bits:

$$(n_1 = \overline{n_2}) \ \wedge \ (n_2 n_3 = 0), \tag{5}$$

which has three satisfying assignments, $(n_1, n_2, n_3) \in \{(1, 0, 0), (1, 0, 1), (0, 1, 0)\}$. To construct a corresponding MWIS representation, we first consider the two MWIS representations for the two involved constraints individually, which are given in Figs. 3(a) and 3(b), and then simply combine them by constructing the union of the individual graphs and add their weights [Fig. 3(d)]. Equivalently, we add the two cost functions of the two individual constraints to obtain the MWIS cost function encoding of Eq. (5):

$$H_{\text{MWIS}} = -\delta(n_1 + 2n_2 + n_3 + n_4) + U(n_1n_2 + n_2n_3 + n_3n_4 + n_4n_2). \quad (6)$$

It is easy to see that ground states of this cost function corresponds to the satisfying assignments in Eq. (5) on the vertices of interest, i.e., vertices 1, 2, and 3.

This example generalizes to the following important observation: consider a set of constraints, $C = \{C_1, C_2, \ldots\}$, allowing for at least one satisfying assignment. Given the MWIS representations of each individual constraint C_i, we can construct a MWIS representation of C by simply adding all the MWIS cost functions for all individual constraints in C. The resulting cost function for C indeed corresponds to a MWIS problem: its graph is simply the union of the individual graphs corresponding to the C_is, with the corresponding weights added on the respective vertices [37].

This is a powerful method that allows us to build MWIS representations of complicated constraints out of simpler ones. We repeatedly make use of this technique in the following sections. The utility of this tool can already be illustrated by noting that the combination of the NOT and the NOR constraints [Fig. 3(a) and 3(c)] is universal. This immediately implies that we can encode any circuit satisfiability problem [38] into a MWIS problem with a constant overhead using this construction.

B. Gadgets for unit-disk transformation

While the representations introduced in the previous subsection allow the encoding of arbitrary constraint-satisfaction problems into MWIS, additional gadgets are required for the specific task of transforming general graphs problems into UDG-MWIS problems, which can then been natively implemented using Rydberg atom arrays. Here, we introduce several particularly useful gadgets in this context.

1. Copy gadget and effective bits

By combining N constraints of the form $n_m = \overline{n_{m+1}}$, we obtain a gadget, called the copy gadget:

$$n_1 = \overline{n_2} = n_3 = \overline{n_4} = \cdots . \quad (7)$$

Here, the information of the bit n_1 is copied to all odd-index bits $n_3, n_5, \ldots$. As the name suggests, this copy gadget is useful in situations where the value of the bit n_1 is needed in several distant locations or in a location in conflict with the unit-disk requirement. Conceptually, copy gadgets "stretch" the representation of a bit from a vertex (a pointlike structure) to a one-dimensional (1D) line, while staying in the paradigm of unit-disk graphs. This technique is similar to other encoding approaches of using wires or chains of virtual vertices [14,18,36,39,40].

Using the techniques developed in Sec. IV A, it is easy to construct the MWIS representation of the copy gadget in Eq. (7). It consists of a one-dimensional graph with N vertices and edges between neighboring vertices. All vertices have a weight 2δ, except for the two boundary vertices of the line, which have weights δ [see Fig. 4(a)]. Indeed,

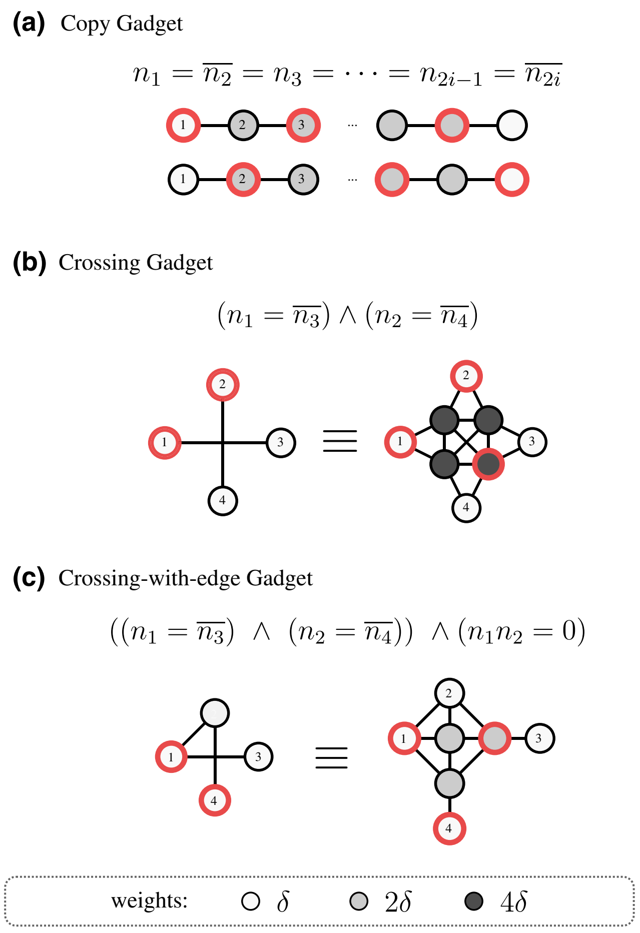

(a) Copy Gadget
$n_1 = \overline{n_2} = n_3 = \cdots = n_{2i-1} = \overline{n_{2i}}$
1
2
3
(b) Crossing Gadget
$(n_1 = \overline{n_3}) \wedge (n_2 = \overline{n_4})$
1
2
3
4
≡
(c) Crossing-with-edge Gadget
$((n_1 = \overline{n_3}) \wedge (n_2 = \overline{n_4})) \wedge (n_1n_2 = 0)$
weights: ○ δ ● 2δ ● 4δ

FIG. 4. Important gadgets for formulating constraint-satisfaction problems as UDG-MWIS. (a) Copy gadget. A 1D line graph encodes an effective bit. The two degenerate MWIS solutions are shown: the subset of odd-numbered vertices (top) and even-numbered vertices (bottom) represent the effective bit values 1 and 0, respectively. In this way, one can copy a single bit to any odd-numbered vertex. (b) Crossing gadget. The four degenerate MWIS solutions of the left graph coincide with the four MWIS solutions on the right graph on vertices 1, 2, 3, 4. One of these solutions is shown. Given a graph that contains a crossing, we can thus replace it with the right UDG, without changing the structure of the MWIS solution. (c) Crossing-with-edge gadget. Similar to (b), we can replace any subgraph of the type depicted on the left with the UDG on the right. One can check the MWIS solutions have one-to-one correspondence. The weights in (a)–(c) are encoded in grayscale according to the legend at the bottom of the figure.

this weighted graph has two degenerate MWIS solutions: $\mathbf{n} = (1, 0, 1, 0, \dots)$ and $\mathbf{n} = (0, 1, 0, 1, \dots)$, corresponding to the two satisfying assignments of Eq. (7). We can thus use these two states to represent the effective binary variable with values 1 and 0, respectively (corresponding to the value of n_1). Note that the 1D line representation does not necessarily need to be drawn as a straight line when embedded in a 2D plane; it can bend and have kinks, as long as the resulting embedding satisfies the unit-disk criterion.

Importantly, we can equip this effective bit with an effective weight w by simply favoring one of the two staggered configurations with respect to the other. For instance, this can be achieved by adding an additional weight w to any one of the equivalent vertices with an odd index in this gadget, e.g., the boundary vertex n_1. More generally, we can induce an effective weight w by any vertex weight configuration as long as it satisfies $\sum_{m=0}^{i} \delta_{2m+1} = w + \sum_{m=1}^{i} \delta_{2m}$ and $0 < \delta_m, w < U$. The latter inequalities ensure that the two staggered configurations remain the two lowest energy states, i.e., states with defect have a lower weight.

2. Crossing gadget

The copy gadget allows the effective representation of a binary variable as a 1D line on a UDG. When there are multiple such variables in a geometric representation, it can be extremely useful to allow two such lines to cross, without introducing any coupling between their corresponding effective degrees of freedom. However, such a crossing manifestly violates the unit-disk constraint. We solve this problem with the crossing gadget.

For this, consider the following set of constraints between four binary variables

$$(n_1 = \overline{n_3}) \ \wedge \ (n_2 = \overline{n_4}). \qquad (8)$$

One way to represented this as a MWIS problem is to identify each variable as an equally weighted vertex in a graph with edges $E = \{(1, 3), (2, 4)\}$. Depending on the relative location of the vertices in the 2D plane (which might be fixed, e.g., due to additional constraints), these two edges might need to cross each other, violating the unit-disk requirement. The crossing gadget is an alternative MWIS representation of the same pair of constraints (8) that avoids this issue. This gadget is depicted in Fig. 4(b) and contains four ancillary binary variables (vertices). Note that the vertices representing the original variables are weighted equally with a weight δ, while the four ancillary vertices have a weight 4δ [41]. This graph is manifestly a UDG and realizes the desired relative geometrical distribution of the vertices 1, 2, 3 and 4. One can easily check that it has four-fold degenerate MWISs, which correspond to the four satisfying assignments on the four original vertices.

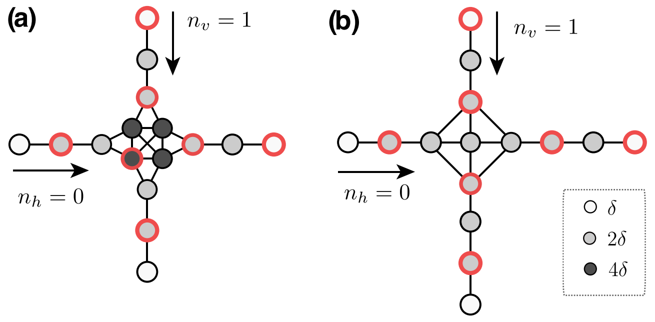


FIG. 5. (a) Two decoupled effective degrees of freedom. By combining the copy gadget and the crossing gadget, we can form two 1D lines, here drawn horizontally and vertically. Both lines represent a binary variable, $n_h \in \{0, 1\}$ and $n_v \in \{0, 1\}$, respectively. The crossing gadget effectively decouples these degrees of freedom. Accordingly, this weighted graph has a fourfold degenerate MWIS, corresponding to the four possible states of two binary variables. One of the MWISs is shown in red corresponding to $n_h = 0$ and $n_v = 1$. Note that each of the four MWISs contains exactly one of the four internal vertices of the crossing gadget, so these internal vertices encode the states of both effective degrees of freedom. (b) Two effective degrees of freedom, defined on a horizontal and a vertical line, respectively, that satisfy an independence constraint $n_h n_v = 0$, introduced by the crossing-with-edge gadget [Fig. 4(c)]. Note that this graph has exactly three degenerate MWIS (out of which, one is shown in red), corresponding to the configurations $(n_h, n_v) \in \{(0, 0), (0, 1), (1, 0)\}$.

In Fig. 5(a), we illustrate how to combine a crossing gadget and the copy gadget to define two decoupled effective binary degrees of freedom living on two lines, a horizontal one and a vertical one. Specifically, in Fig. 5(a), the two effective degrees of freedom are realized by the two staggered configurations of the horizontal and vertical line, and the crossing gadget decouples them; this structure is realized by extending each boundary vertex of the crossing gadget using the copy gadget. Following the recipe given in Sec. IV A 2, one defines a vertex weight pattern with weights 4δ on the interior vertices of the crossing gadget, weight 2δ on the exterior vertices of the crossing gadget and on all vertices of the lines, except for the boundary vertices of each line, which have a weight δ.

By generalizing this example, we can see that the copy gadget allows us to represent binary variables as lines and the crossing gadget allows these lines to cross without introducing any interactions or constraints between their effective degrees of freedom, so we can arrange these effective 1D lines arbitrarily in 2D without worrying about crossings between them.

3. Crossing-with-edge gadget

The crossing gadget is useful to decouple effective degrees of freedom defined on lines, even if the lines cross. In contrast, we now introduce a gadget that allows us to introduce a specific type of interaction between the effective degrees of freedom. Specifically, we are interested in a gadget that introduces the independence constraint $n_u n_v = 0$ between two effective variables, n_u and n_v, when their corresponding lines cross. For this, we consider the situation where four binary variables must satisfy the constraints

$$((n_1 = \overline{n_3}) \wedge (n_2 = \overline{n_4})) \wedge (n_1 n_2 = 0), \tag{9}$$

where in this case, $n_u \equiv n_1$, $n_v \equiv n_2$. This corresponds to the MWIS problem on the graph on the left of Fig. 4(c). In particular, we consider the situation where the vertices are geometrically positioned relative to each other in a way that requires a crossing; this case indeed occurs when n_1, n_3 and n_2, n_4 each belong to a line (created by a copy gadget) that corresponds to the effective binary variable associated with n_1 and n_2. This graph is, however, not a unit-disk graph, so we introduce the crossing-with-edge gadget shown on the right of Fig. 4(c). The resulting graph is manifestly a unit-disk graph with a threefold degenerate MWIS solution, corresponding to the three satisfying assignments of the crossing-with-edge constraint required in Eq. (9).

Analogous to the discussion of the crossing gadget, we can also combine the crossing-with-edge gadget with copy gadgets to obtain two crossing lines [drawn horizontally and vertically in Fig. 5(b)] that host two effective binary degrees of freedom, respectively, with an independence constraint between them. The resulting weight pattern is shown in Fig. 5(b).

V. ARBITRARY CONNECTIVITY

Using the suite of encoding gadgets introduced in the previous section, we can now encode a variety of computational problems into UDG-MWIS, which can then be readily implemented on Rydberg atom arrays. Here, in this section, we discuss three example applications in detail: MWIS on graphs with arbitrary connectivity, QUBO problem, and the integer factorization problem. As we will see later, the resulting UDGs can be embedded on a square lattice with at most a quadratic overhead. The recipe involves two main steps: the first is to construct the so-called *crossing lattice* using the copy gadget, and the second is to apply crossing replacements to encode arbitrary connectivity.

A. The crossing lattice

Consider an optimization problem for N binary variables, such as the MWIS problem defined on an arbitrary graph $G = (V, E)$, where each vertex represents a binary degree of freedom. To realize arbitrary connectivity, we first use the copy gadget to represent each vertex $v \in V$ by a 1D vertex line. The state of the binary variable associated with a vertex v (0 or 1) can then be accessed at any odd-index vertex of the corresponding line [Fig. 4(a)]. As detailed below, interactions between the effective degrees of freedom represented by these lines can be introduced at points where the lines cross. To achieve arbitrary connectivity, each line must thus cross every other line at least once. A simple layout achieving this is shown abstractly in Fig. 2(b), where each line is drawn with a vertical and a horizontal segment, forming an upper triangular crossing lattice. In this way, a line (representing vertex v) crosses any other line (representing vertex w) exactly once. At these crossing points, we can then use the various crossing gadgets introduced in the previous section to induce interactions between v and w or to keep them decoupled. This is detailed concretely in Secs. V B and V C below. In addition to introducing interactions, these gadgets also turn the resulting graph explicitly into a UDG. Note that the resulting graph can be constructed by $N(N-1)/2$ "tiles," each containing eight vertices for a tile formed by a crossing gadget, or seven vertices for a tile formed by a crossing-with-edge gadget. Taking into account also the boundary vertices, we conclude that this construction leads to a UDG with at most $4N^2$ vertices, corresponding to the optimal quadratic overhead for arbitrary connectivity [22]. We note that this particular choice of "weaving" lines together may be suboptimal, especially if the connectivity is sparse. In such a case, the total size of the lattice and thus the encoding overhead can be reduced by forming more sophisticated crossing lattices. More details of the simplification steps are included in Appendix A.

B. Maximum-weight independent set

Building on the above recipe, we now detail how to map the MWIS problem on an arbitrary weighted graph $G = (V, E)$ with vertex weights w_v ($v \in V$) to a UDG-MWIS problem. As shown in Fig. 6(a), we first create a crossing lattice using the copy gadget. At each crossing point between two lines (corresponding to vertices $u, v \in V$), we decouple the effective degrees of freedom using a crossing gadget if $(u, v) \notin E$, or induce an independence constraint for the effective degrees of freedom via a crossing-with-edge gadget if $(u, v) \in E$, as shown in Fig. 6(b). We note that this results in a UDG that can be embedded on a square lattice (whose diagonal sets the unit-disk radius). The weights on the vertices of this UDG are 2δ on all vertices except the four interior vertices of the crossing gadget (which have weight 4δ) and the two boundary vertices of each line, whose weights are chosen to introduce the correct effective weights w_v for the effective variable represented by each line. We choose a convention in which the first vertex along the line v has a weight $\delta + w_v/2$, and the last vertex along this line has a weight $\delta - w_v/2$. Note that, to guarantee that the MWIS of the resulting UDG is indeed formed by the proper low-energy configurations of each line, the weights have to satisfy $2w_v \leq \delta$. This can always

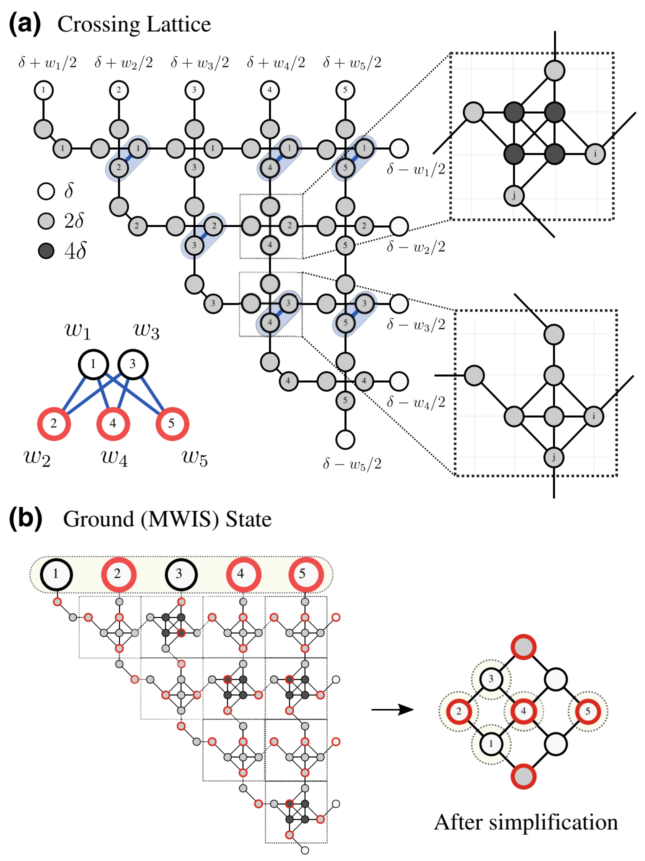


FIG. 6. Example encoding procedure for the MWIS problem on the $K_{2,3}$ (non-unit-disk) graph into UDG-MWIS. (a) Crossing lattice. Each vertex v is represented by a line using the copy gadget, with odd-numbered vertices labeled. Each line is bent to form a triangular lattice, giving a crossing point between any two variables. If the two variables share an edge in the original graph, we draw an edge between their representative vertices on the lattice. A UDG is then obtained by replacing each crossing with a crossing or crossing-with-edge gadget. (b) UDG representation and the corresponding ground-state solution. Each crossing in (a) is replaced with a unit cell containing at most eight vertices, thus resulting in a final mapping with 92 vertices. The MWIS of the original graph ($\{2,4,5\}$) can be read out from the boundary vertices of the ground state of the mapped graph. This mapping can further be simplified (see Fig. 10) to a mapping of only nine vertices, where the vertices corresponding to the original degrees of freedom still encode the desired MWIS solution.

be achieved by a proper normalization of the weights, or a suitable choice of δ (for more details see Appendix B).

The MWIS of the mapped problem can be straightforwardly transformed back to a valid solution in the original problem. Indeed, since the state of the effective degrees of freedom associated with each line can be accessed at the first vertex of the line, the MWIS of the original problem is directly given by the configuration of the boundary vertices of the MWIS of the mapped problem. This solution readout is shown as the yellow ellipses of Fig. 6(b).

C. Quadratic unconstrained binary optimization

QUBO is a paradigmatic NP-hard combinatorial optimization problem that has a wide range of applications. Generally, it seeks to find an input configuration that minimizes a quadratic polynomial function

$$f(z) = \sum_{i<j} J_{ij} z_i z_j + \sum_i h_i z_i, \qquad (10)$$

where the domain of f is binary bitstrings $z \in \{\pm 1\}^N$. QUBO is also called the Ising problem, where each bit can be represented by a spin 1/2 degree of freedom, and the QUBO solutions correspond to the ground states of the Ising model.

To encode the QUBO problem in a UDG-MWIS, we again start with constructing the crossing lattice, with each line encoding one of the binary variables z_i [Fig. 7(a)]. For simplicity, we choose the number of vertices along each line to be even. We then use the crossing gadget at each of the crossing points of the lattice, which decouples all the N effective binary degrees of freedom. Recall that at this point in the construction all vertices in the resulting graph have a weight 2δ, except for the boundary vertices on each line, which have a weight δ and the four interior vertices of each crossing gadgets, which have a weight 4δ. The QUBO cost function is then imposed on the effective degrees of freedom by adjusting these weights as follows: Firstly, the weight of the two boundary vertices of line i is adjusted to $\delta + w_i$ for the first vertex and $\delta - w_i$ for the last one [see Fig. 7(a)]. It is easy to see that for lines of even length this induces the linear term $h_i z_i$ for the effective degree of freedom z_i, with $w_i = h_i$ up to normalization (see Appendix B). Secondly, the weights of the internal four vertices of the crossover gadget between the lines representing the bits i and j are adjusted to $4\delta \pm w_{ij}$ as depicted in the inset of Fig. 7(a), where $w_{ij} = J_{ij}$ up to normalization. To see that this induces the quadratic interaction term $J_{ij} z_i z_j$ between the two effective bits, recall that exactly one of the four ancillary vertices of the crossing gadget is part of the MWIS, and that this vertex is determined by the configuration of the effective degrees of freedom z_i and z_j [see Figs. 5(a) and 7(b)]. Similar to the MWIS encoding, the additional weights have to be appropriately normalized, such that the ground state of the cost Hamiltonian consists of configurations that correspond to valid (i.e., defect-free) configurations of the effective degrees of freedom. This is guaranteed by a normalization such that $\max_i(\sum_j |w_{ij}|, |w_i|) < \delta$. For more details on the normalization, see Appendix B. Similar to the MWIS problem, the ground state of the QUBO problem can be directly inferred from the ground state of the resulting UDG-MWIS problem. An example is shown in Fig. 7(c) highlighting the MWIS for a random choice of J_{ij} and h_i, and $N = 5$. In the inset, we confirm that the MWIS state is indeed the one corresponding to the solution

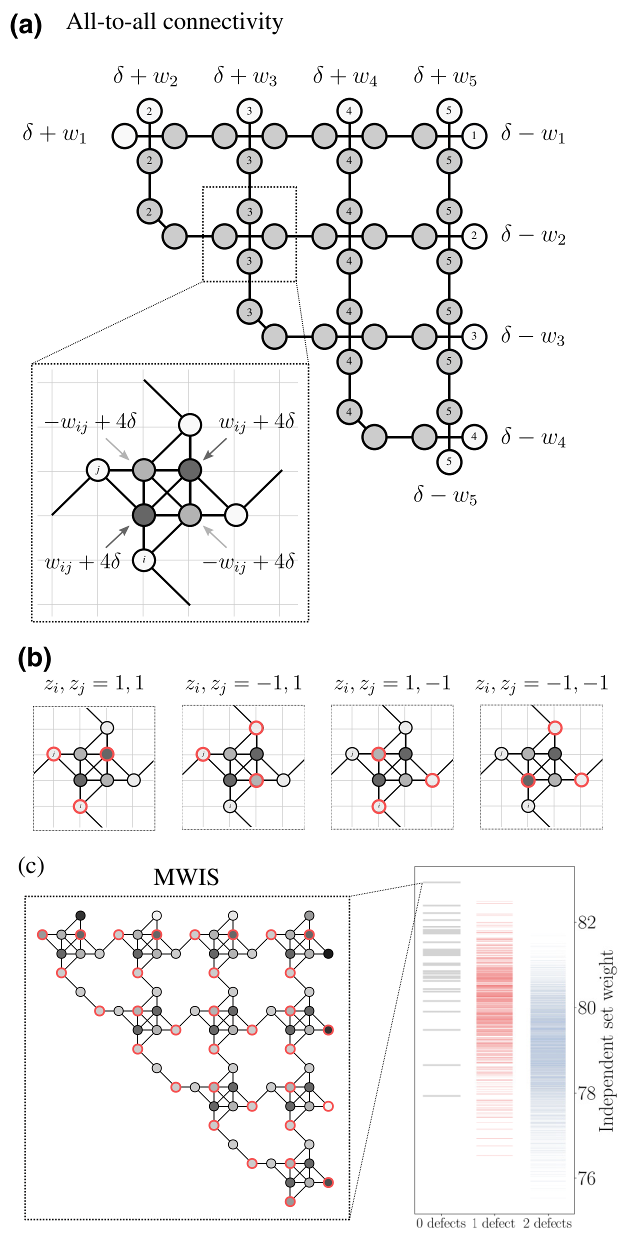


FIG. 7. Example encoding procedure for the QUBO and Ising problem for a 5-bit system. (a) Crossing lattice. Similar to the MWIS mapping, we can construct the UDG-MWIS representation of a generic QUBO problem by inserting a gadget at each crossing. The gadget has a similar structure as the crossing gadget used in the MWIS encoding, but the weights on the ancillary vertices are biased to induce quadratic interaction terms w_{ij} between the effective degrees of freedom; see (b). (c) Example of an encoded $N = 5$ QUBO problem, and the high-weight spectrum of the encoded cost function, illustrating that the MWIS is indeed in the 0-defect sector and thus encodes the solution of the QUBO problem. The QUBO solution $\{-1, +1, +1, +1, +1\}$ is encoded in the boundary of the graph.

of the QUBO problem. We also show the weights of other independent sets, including those that do not correspond to valid configurations of the effective degrees of freedom, i.e., configurations that include defects. We note that the weights of the configurations in the zero-defect sector have a one-to-one correspondence with the spectra of the original QUBO problem. One can see that some states with defects have a higher total weight than some states that represent valid configurations of the effective variables (i.e., without defects), but, importantly, the MWIS is guaranteed to be in the zero-defect sector given proper normalization.

In summary, any QUBO problem on N variables can be encoded in a UDG-MWIS problem with at most $4N^2 + \mathcal{O}(N)$ vertices. For restricted connectivity, one may construct a lower-overhead crossing lattice; for details, see Appendix C.

D. Integer factorization

As a final example, we now formulate the problem of decomposing an n-bit composite integer $m = pq$ into its prime factors p and q as the UDG-MWIS problem. To this end, we use the binary representation for the integer $m = \sum_{i=0}^{n-1} 2^i m_i$, with $m_i \in \{0, 1\}$, $p = \sum_{i=0}^{k-1} 2^i p_i$ for the k-bit integer, and $q = \sum_{i=0}^{n-k-1} 2^i q_i$ for the $(n-k)$-bit integer. The factoring problem thus amounts to finding the unknown bits p_i and q_i such that

$$\sum_{i=0}^{n-1} 2^i m^i = \sum_{i=0}^{k-1} \sum_{j=0}^{n-k-1} 2^{i+j} p_i q_j . \tag{11}$$

Note that k is *a priori* unknown. However, this is not an issue, as one can consider the problem (11) for each possible value $k = 1, 2, \ldots, n/2$, or, alternatively, consider n-bit representations of both p and q.

We proceed by introducing ancillary binary variables $s_{i,j}$ and $c_{i,j}$, which may be interpreted as partial sum bits and carry bits, respectively. Using elementary algebra, the factoring problem (11) may be expressed as a system of $k(n-k)$ coupled equations [42,43]

$$s_{i,j} + 2c_{i,j} = p_i q_j + s_{i+1,j-1} + c_{i-1,j} \tag{12}$$

for $i = 0, \ldots, k-1$ and $j = 0, \ldots, n-k-1$, where the values $c_{-1,j} = c_{i,0} = s_{i,-1} = 0$ are fixed and we identify $s_{0,j} = m_j$. Factoring m thus reduces to finding binary values for $s_{i,j}$, $c_{i,j}$, p_i, and q_j such that the $k(n-k)$ Eq. (12) are all satisfied.

To embed this system of equations in a 2D plane, we use the copy gadget to copy the values of p_i and q_j to new ancillary variables $p_{i,j}$ and $q_{i,j}$ with the copy constraints

$$p_{i,j} = p_{i+1,j}, \tag{13}$$

$$q_{i,j} = q_{i,j+1}, \tag{14}$$

and identify $p_i \equiv p_{i,0}$ and $q_j \equiv q_{0,j}$. We then can write Eq. (11) as

$$s_{i,j} + 2c_{i,j} = p_{i,j} q_{i,j} + s_{i+1,j-1} + c_{i-1,j} . \tag{15}$$

We refer to the three Eqs. (13)–(15) (for a given i,j) as the factoring constraints $F_{i,j}$. The constraints $F_{i,j}$ are manifestly local in two dimensions, in the sense that the variables $s_{i,j}$, $c_{i,j}$, $p_{i,j}$, and $q_{i,j}$ can be arranged on a square lattice such that all factoring constraints $F_{i,j}$ involve only neighboring or diagonally neighboring variables. A graphical representation of this is given in Fig. 8(a), with the box at lattice point (i,j) representing the constraints $F_{i,j}$.

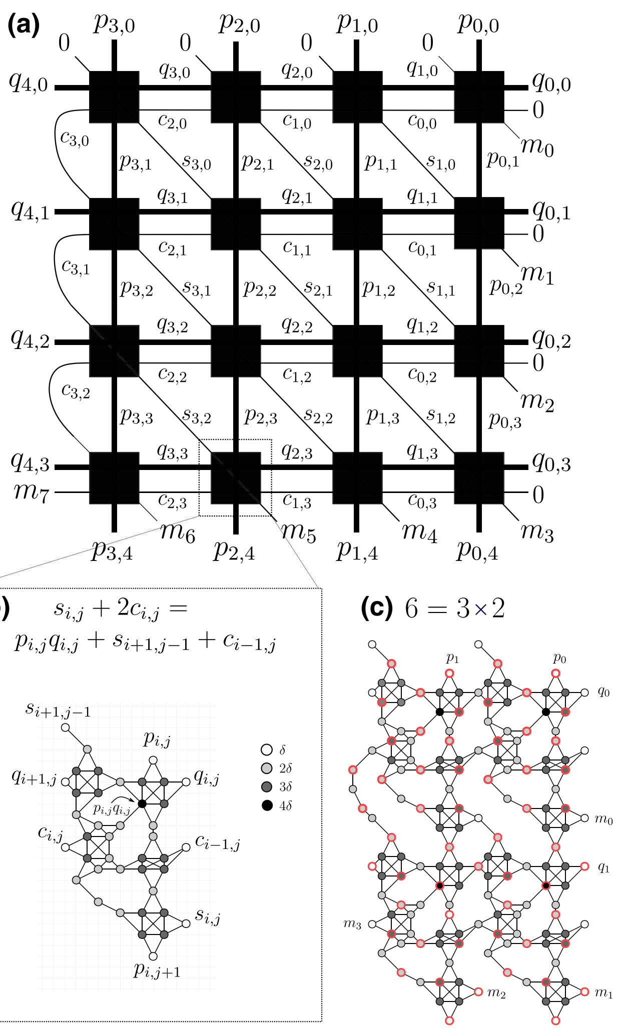


FIG. 8. Encoding procedure for integer factorization. (a) Graphical representation of the set of equations to be satisfied for integer factorization ($m = p \cdot q$). The factor bits p_i, q_i and binary variables $s_{i,j}$, $c_{i,j}$ are represented by copy lines to construct an effective square crossing lattice for the problem, with a filled square at crossings and the integer bits m_i specifying some of the boundary conditions of the problem. (b) Each filled square represents a set of equality constraints between the binary variables associated with the adjacent legs. The final UDG-MWIS can be obtained by replacing each square with the factoring gadget that enforces the mathematical constraints relevant for the factoring problem. The unit-disk radius should be slightly larger than $2\sqrt{2}$ times the lattice constant. (c) An example of the UDG-MWIS representation of the factoring problem $6 = 3 \times 2$.

Specifically, each line represents a binary variable, and each box represents the factoring constraints that have to be satisfied by the variables connected to it. We note that each variable enters in exactly two factoring constraints, except at the boundary.

This formulation of the factoring problem allows for a mapping to a UDG-MWIS problem. Specifically, we introduce a new factoring gadget consisting of a weighted 32-vertex unit-disk graph depicted in Fig. 8(b), where we identify eight of the vertices with the eight variables involved in the factoring constraints $F_{i,j}$. See Appendix D for more details on the construction of the factoring gadget. In particular, we follow the design principle given in Sec. IV A: the factoring gadget is designed such that (i) the MWIS space is degenerate, (ii) every MWIS coincides with a valid solution of $F_{i,j}$ on the vertices that represent the involved variables, and (iii) every valid solution of $F_{i,j}$ is represented by at least one MWIS. All these requirements can be checked by exhaustive search for the factoring gadget depicted in Fig. 8(b). Since each variable has to satisfy two factoring constraints [see Fig. 8(a)], we design the factoring gadget such that this geometrical requirement can be easily met. Indeed, we can represent the full set of constraints $\{F_{i,j} | i = 0, \ldots, k-1; j = 0, \ldots, n-k-1\}$ as a unit-disk graph (of unit radius $r = 2\sqrt{2}$ on a square lattice), by repeating the factoring gadget on a $k(n-k)$ square lattice, as depicted in Fig. 8(c). This construction therefore results in a lattice with some of the boundary conditions fixed by the values of m_i, such that the MWIS of the rest of the graph reveals the values of p_i and q_j, satisfying Eq. (11), thus providing the solution for the factoring problem.

VI. NUMERICAL SIMULATIONS

In previous sections, we demonstrate an encoding strategy to map a computation problem with arbitrary connectivity onto a maximum-weighted independent set problem on a unit-disk graph, showing that the ground state of the mapped problem encodes the solution of the original problem. We now present numerical simulations of quantum algorithms to demonstrate the impact of the proposed mapping procedure on quantum performance. Specifically, we use the quantum adiabatic algorithm and consider the MWIS problem on an ensemble of non-isomorphic, simply connected six-node graphs, which includes three non-unit-disk graphs. For simplicity, we choose uniform weights for each graph $\Delta_v = \Delta$. Using the procedure introduced in Sec. V B and further simplification steps described in Appendix A, we map each of the original MIS problems to a UDG-MWIS problem and compare performance metrics of QAA for both problems. There are a total of 112 non-isomorphic graphs with six vertices. We sample 54 such graphs: due to limits of classical simulation, we evaluate only instances whose mapped graphs contain no more than

25 vertices; we also do not consider the 40 graphs that can be directly embedded as UDGs on the square lattice without any overhead.

Figure 9 presents the performance results for the ensemble of graphs. The QAA for the MWIS problems may be performed for a particular graph by varying $\Delta_v(t)$ and $\Omega_v(t)$ in the Rydberg Hamiltonian (1) [13]. Typically, the QAA is designed by initializing all qubits in the $|0\rangle$ state, where $\Delta(t=0) < 0$ and $\Omega(t=0) = 0$ (with $U > 0$). QAA for MWIS is usually done in two or three stages [10]. First, $\Omega(t)$ is ramped up to a nonzero value while $\Delta(t)$ is slowly tuned from negative to zero. Next, $\Omega(t)$ is ramped off while $\Delta(t)$ is slowly tuned from zero to positive. In this way, the initial state is adiabatically connected to the ground state of the final, classical Hamiltonian whose ground state encodes the MWIS solution of the UDG. For sufficiently slow ramps, the quantum state of the system follows the instantaneous ground state of the time-dependent Hamiltonian and thus the final state corresponds to the MWIS of the graph.

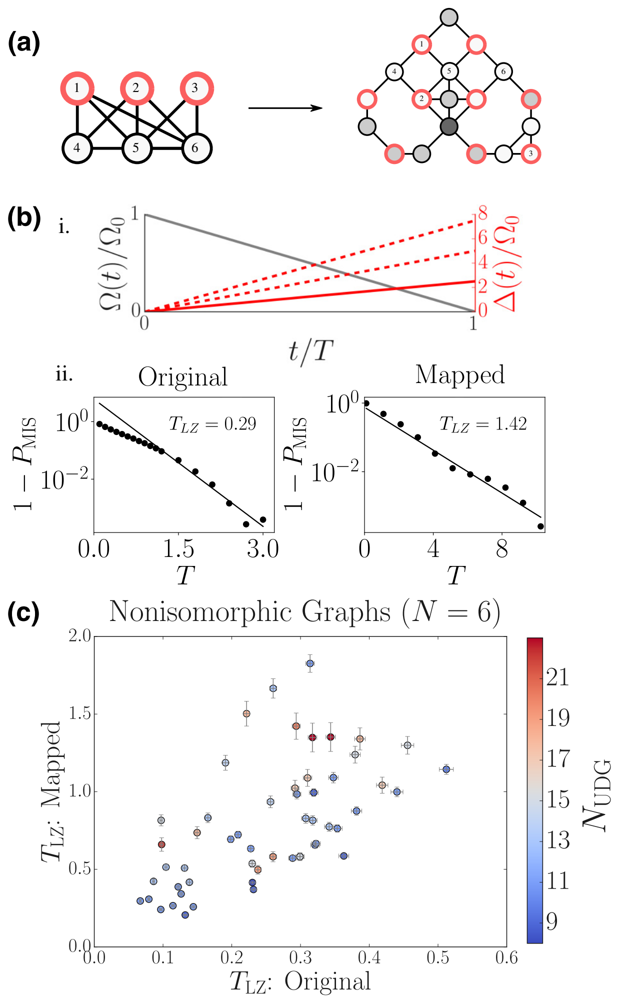


FIG. 9. QAA performance for MWIS on original and mapped graphs. (a) Example unit-disk mapping and MIS configuration of a six-vertex graph. The original graph (left) is unweighted, and the mapped graph is a weighted UDG (right). The corresponding vertices are labeled with numbers. (b) i. Rabi frequency $\Omega(t)$ and detuning $\Delta(t)$ sweep used in the QAA protocol. The global detuning $\Delta(t)$ is shown in solid red, while $2\Delta(t)$ and $3\Delta(t)$ are shown in dashed lines. ii. Computed P_{MIS} for the original and mapped graphs as a function of total sweep time T. The adiabatic timescale T_{LZ}, which is related to the minimum energy gap, is extracted from the long-time Landau-Zener fitting. (c) Scaling of T_{LZ} for mapped graphs for the ensemble of unweighted nonisomorphic graphs with size $N = 6$.

In order to focus on adiabatic behavior near the gap closing point and account for the weights in the MWIS problem, we choose to initialize the state of the system starting at the end of the first stage, in the ground state of the Hamiltonian $\Omega(t=0) = \Omega_0$ and $\Delta(t=0) = 0$. Then, Ω is linearly ramped to zero over a time T and each of the detunings are linearly ramped from zero to a final value. Note that because the weights Δ_v on each vertex may be different, the rate of change of detuning may be different on each vertex. This protocol is shown in Fig. 9(b).

For our numerical simulation, we work in the limit of $\Delta_0, \Omega_0 \ll U$, where the nonindependent set space of the graph can be neglected (in experiments, this corresponds to the strong Rydberg-blockade limit). In this limit, we can restrict the simulation to the independent set subspace of the graph. We also ignore long-range Rydberg interactions, since the original graph does not have a geometric description.

The performance of QAA is often discussed via an analysis of the minimum spectral gap along the parameter path. However, the minimum gap alone is not sufficient to understand the time scales for adiabaticity, as the structure of matrix elements between ground and excited states can cause larger or smaller diabatic effects. Furthermore, it can be ambiguous for instances where multiple degenerate ground states exist. We therefore compare the QAA performance on the original and mapped problems by directly comparing their adiabatic time scales. Specifically, we evaluate the adiabatic time scale by extracting a Landau-Zener time scale, T_{LZ}, which is the characteristic time needed to evolve the system adiabatically. T_{LZ} is determined by fitting numerical results to the expected long-time behavior of the ground-state probability $P_{\text{MIS}} = 1 - e^{a-T/T_{\text{LZ}}}$ at T where $P_{\text{MIS}} \gg 0.99$. This procedure is described in more detail in Ref. [13].

Figure 9(c) presents results of this analysis, comparing the extracted Landau-Zener time scale for the original graphs in our ensemble with the Landau-Zener time scale of the corresponding mapped UDGs. The simulation results indicate that for the graphs considered, the timescale for adiabaticity of a mapped MWIS problem is correlated with that of the original problem: the correlation appears to be linear, but the limited range of data precludes a reliable fitting; the spread of the data is likely due to the specific structure of the graph, but there is no clear dependence on the number of vertices in the mapped graph. Unfortunately, these instances are far from the large-problem size limit and the displayed time scales give us little intuition about the asymptotic performance of the quantum algorithm for larger graphs. To have more conclusive understandings of the performance of quantum algorithms, one has to study it in Rydberg atom array experiment on larger graphs.

VII. CONCLUSIONS AND OUTLOOK

In this work, we described an encoding strategy to map a variety of computation problems with arbitrary connectivity to maximum-weighted independent set problems on unit-disk graphs, which have a hardware-efficient implementation of quantum optimization on neutral, trapped atoms interacting via Rydberg states. The encoding incurs at most a quadratic overhead in the number of variables in the optimization problem and is thus very efficient. In addition, the mapping follows an explicit, straightforward procedure, which produces a unit-disk graph that can be embedded on a square lattice with favorable conditions on the necessary unit-disk radius and hence enables practical implementation with Rydberg atom arrays.

We provided three concrete examples for the problem mapping: MWIS problem on general graphs with arbitrary connectivity, the QUBO problem, and the integer factorization problem. In all examples, we show how the formation of a crossing lattice together with a few encoding gadgets enable a simple, unified approach to map a wide range of computation problems into UDG-MWIS. Numerical simulations indicate that the performance of quantum algorithms on the mapped problems is directly correlated with that of the original problems. If the linear correlation in the Landau-Zener time persists asymptotically, this suggests that any quadratic quantum speedup [10] in the original graph may be transferred to an equivalent speedup on the unit-disk graph.

While in this work we focused on encodings into UDG-MWIS, we remark that similar strategies can also be designed for encoding computation problems into unweighted UDG-MIS problems. This is favorable for experimental implementation when local detuning capabilities are not available. The details for unweighted encoding are described in a companion paper [22]. There are at least several interesting future directions that deserve further explorations. First, our approach may be generalized to encode computation problems that include interactions involving three or more variables, such as higher-order unconstrained binary optimization (HUBO) problems; the NOR gate shown in Fig. 3(c), for example, is a constraint that involves three variables. It would also be interesting to consider encoding approaches beyond 2D geometry, for example, by generalizing the idea of a crossing lattice to a "crossing cube" in three dimensions (3D); one may design encoding methods with lower overhead in 3D or make use of the third dimension in other ways such as thinking of it as the time direction for circuit satisfiability problems. Finally, we emphasize that our encoding strategy focuses on exact (ground-state) solutions. An interesting question is to what extent such strategies can be employed for approximate optimization. It will be important to understand the effects of the encoding mappings on the performance of quantum algorithms in terms of excitations into higher-energy states.

The source code that implements the mappings in this work is available in the GitHub repository UnitDiskMapping.jl. The (sub)optimal configurations and weight spectrum of the maximum independent set problem instances are computed using the Julia package GenericTensorNetworks.jl, which implements the generic tensor-network algorithm [44].

ACKNOWLEDGMENTS

We thank Leo Zhou, Soonwon Choi, Xiu-Zhe (Roger) Luo, Harry Levine, Giuliano Giudici, Zhongda Zeng, Peter Zoller, and Nathan Gemelke for helpful discussions. We acknowledge financial support from the DARPA ONISQ program (Grant No. W911NF2010021), the Army Research Office (Grant No. W911NF-21-1-0367), the Center for Ultracold Atoms, the National Science Foundation, the Vannevar Bush Faculty Fellowship, the U.S. Department of Energy (DE-SC0021013 and DOE Quantum Systems Accelerator Center (Contract No. 7568717), the ERC Starting grant QARA (Grant No. 101041435), and an ESQ Discovery Grant. Jin-Guo Liu acknowledges funding support provided by QuEra Computing Inc. through a sponsored research program.

APPENDIX A: OVERHEAD REDUCTION

In this Appendix, we provide additional strategies to the introduced mapping scheme to further reduce the overhead required for encoding the computation problems into UDG-MWIS problems. We introduce several simplification techniques that can be easily automated to reduce the overhead of the final mapped graph, allowing us to map specific graphs with significantly less overhead.

1. Crossing lattice reduction

a. Pathwidth reduction

As discussed in the main text, to impose interaction constraints for arbitrary connectivity, we construct a crossing lattice. We can reduce the depth of the crossing lattice by reordering the vertices, thus allowing the final mapping to scale with the pathwidth of the original graph. A graph $G = (V, E)$ has a pathwidth $pw(G) \leq k$ if and only if it has a vertex order $v_1, v_2, \ldots, v_n$ such that for any $1 \leq i \leq n$, there are at most k vertices among $\{v_1, \ldots, v_i\}$ that have neighbors in $\{v_{i+1}, \ldots, v_n\}$. We can obtain $pw(G)$ with a path decomposition. A path decomposition is a sequence of "bags" $(X_1, X_2, \ldots, X_N)$, where $X_i \subseteq V$ such that

$$v \in X_i,\ v \in X_k \Longrightarrow \forall j \in [i,k], v \in X_j. \qquad \text{(A1)}$$

In other words, every vertex $v \in V$ in G belongs to at least one bag and the set of bags containing v forms a connected interval of the sequence $(X_1, X_2, \ldots, X_N)$. Moreover, for each edge $e \in E$, there is a bag X_i that contains both endpoints. We define the width w of a path decomposition as the maximum size of the bags and pathwidth $pw(G) = w - 1$.

This is advantageous because for a sparse graph, the pathwidth is usually much smaller than the number of vertices. For example, the pathwidth of a 3-regular graph is asymptotically bounded by $n/6$, and the pathwidth of a tree graph is logarithmic in n. By inspecting the appearance of the order of vertices in a bag in an optimal path decomposition, we get a good vertex reordering that reduces the size of the crossing lattice as described below.

b. Vertex reordering

One can reorder the vertices in the encoding mappings to reduce the depth of the final mapped graph to the pathwidth of the graph, i.e., the size of the crossing lattice is thus $O(N * pw(G))$. More concretely, we can reduce the overhead in the mappings by minimizing the length of the copy lines and reducing the number of crossing gadgets needed. Reordering the vertices allows us to cluster crossings in the crossing lattice where the two degrees of freedom interact (such as when two vertices share an edge in the MWIS problem), and thus reduce the number of unnecessary crossings in the crossing lattice.

Graphically, as seen in Fig. 10, for the example of the MWIS problem on the $K_{2,3}$ graph. Figure 10(a) depicts the original crossing lattice discussed in Sec. V A. Reordering the vertices according to the path decomposition of the graph [Fig. 10(b)] reduces the number of empty squares (crossing gadgets) from 4 to 2, thus reducing the mapping overhead even when $pw(G) + 1 = N$; in general, we have $pw(G) + 1 < N$ for most graphs. Because the crossing lattice is a 2D mapping, we can apply the same strategy to reorder vertices along both axes: one can find a bipartition of a graph to construct a crossing lattice that minimizes the number of unnecessary crossings (or empty squares), as shown in Fig. 10(c). Using vertex reordering, for the $K_{2,3}$ example graph, we can construct a simplified unit-disk mapping of nine vertices [Fig. 10(d)] whereas the direct mapping has 92 nodes.

Thus, we can generally simplify the standard mapping and reduce the overhead by restructuring the crossing lattice to reduce the length of copy lines and minimize unnecessary crossings. The optimal vertex reordering requires computing the optimal path decomposition of a graph,

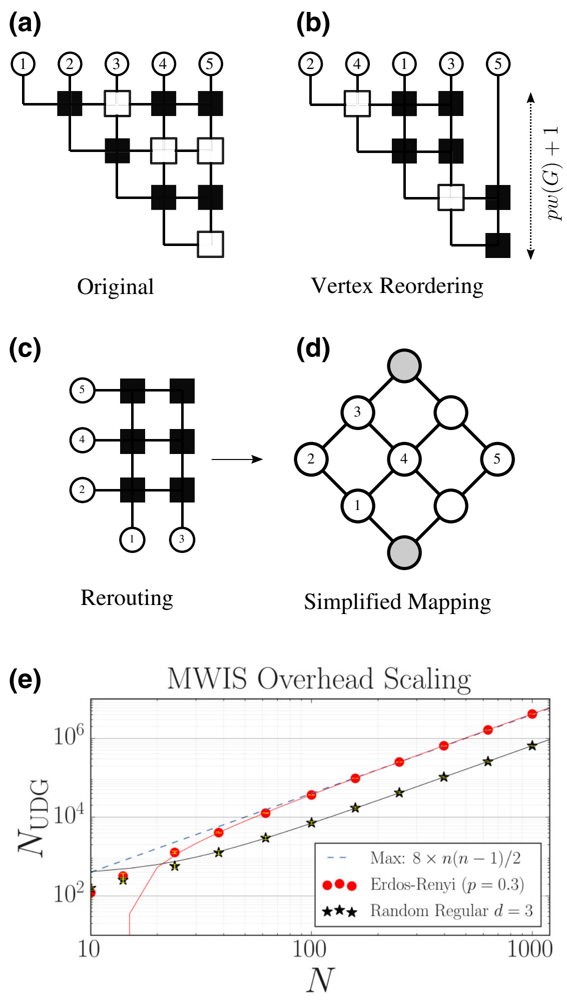
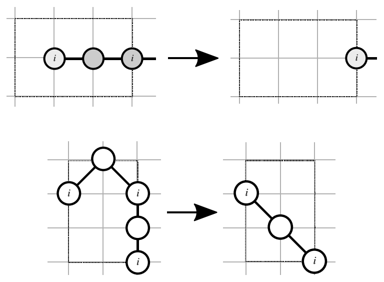


FIG. 10. (a)–(c) Simplification techniques to reduce the overhead. Vertex reordering can be used to reduce the depth of the crossing lattice. (d) Final mapping of the $K_{2,3}$ graph after bipartition and vertex reordering, reducing the overhead mapping from 92 nodes to 9 [see Fig. 6(b)]. (e) MWIS overhead scaling as a function of the number of vertices of original randomly generated graphs for chosen graph classes using a greedy pathwidth reduction algorithm.

which is itself an NP-hard problem. For small graphs, the optimal path decomposition can be effectively computed with the branching algorithm [45]; for larger graphs, one can use heuristic algorithms to find good path decomposition. This strategy allows us to achieve an overhead scaling for a chosen set of graph classes shown in Fig. 10(e). A more detailed discussion of how vertex reordering can reduce the number of crossings is included in the companion paper [22].

2. Simplification gadgets

We can further reduce the mapping overhead from the standard encoding procedure, or any valid unit-disk mapping by introducing rewriting rules, or gadgets that maintain the integrity of the mapping, while also reducing the overhead of the graph. Simplification gadgets are most useful for the MWIS problem, where node weights are more uniform, but simplification gadgets should preserve the weight constraints of the original problem. For example, here are some simplification rules:



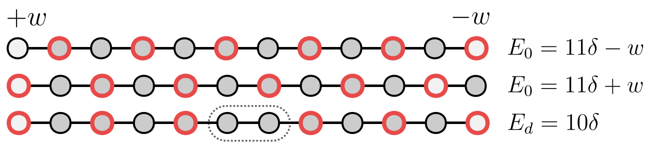
APPENDIX B: DEFECTS AND MWIS GUARANTEES

As described in the main text, one needs to be careful with the proper normalization of the vertex weights to ensure the ground state of the mapped problem correctly encodes the solution of original problem. In this Appendix, we provide more details on normalization for the MWIS and QUBO problem.

For the MWIS mapping shown in Fig. 6, the UDG-MWIS problem is guaranteed to encode a valid solution of the original problem only if the additional weights (biases) are properly chosen. If a bias is too large, it may be energetically favorable to violate a constraint in the problem, causing the MWIS to be an invalid solution. These constraint violations, in the context of the copy gadget, are called "defects." It is thus imperative to limit the size of the biases to guarantee that the MWIS is a valid solution.

For the constraint-satisfaction problems constructed as reductions for MWIS and QUBO, there is a hierarchy of constraints. At one scale is δ, which corresponds to the energy scale of the unweighted problem and at another scale is the linear (w_i) and quadratic (w_{ij}) biases that prefer certain MWIS and QUBO solutions for the weighted problems.

The safest normalization of biases is to constrain that the total weights is less than the cost of a single constraint violation. For the copy gadget, which encodes the constraint $(n_1 = \overline{n}_2) \wedge (n_2 = \overline{n}_3) \wedge \cdots$, the cost of violating one constraint and adding one defect is at least δ. For instance, consider a length-12 copy gadget with bias w



For the configuration with a defect (the third line), the left side incorrectly represents a 1, while the right side represents a 0. Similarly, for the crossing gadget, removing the vertex from a clique costs an energy of at least 2δ, at the expense of adding two defects; one to the horizontal and one to the vertical copy gadget. Thus, the most conservative normalization of biases, which sums over all clauses, is

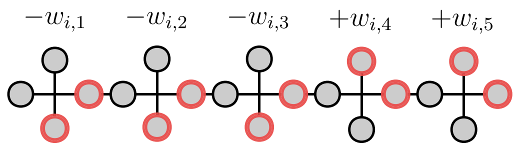
$$\delta > \sum_{ij} |w_{ij}| + |w_i|. \tag{B1}$$

Unfortunately, such a normalization is too conservative. For the MWIS problem, the normalization goes as $1/N$, while, for the QUBO problem, the normalization goes as $1/N^2$. A larger bias that still guarantees a valid MWIS can be found by inspecting the structure of the copy gadget and crossing lattice.

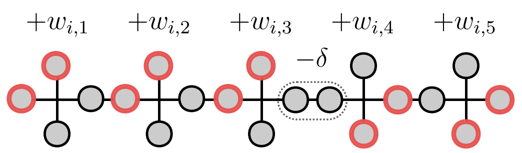
For the MWIS problem, it is beneficial to inspect only a single copy gadget. Consider a copy gadget of length $2n$ and biases $+w$ on one end and $-w$ on the other. The valid solutions have energies $(n-1)\delta \pm w$, and the single-defect invalid solution has an energy $n\delta$. Thus, in order to guarantee a valid solution, it serves to have every bias $\delta > w_i$. In this way, the normalization must obey the constraint

$$\delta > \max_i |w_i|. \tag{B2}$$

For the QUBO problem, it is similarly beneficial to inspect only a single copy gadget. Single-defect solutions to the copy gadget may potentially be energetically favorable if the cost of adding a defect is outweighed by satisfying more quadratic terms. As an extreme case, consider a bit i in a state -1, with a QUBO interaction $w_{ij} > 0$ with every other qubit j. However, suppose the optimal state of every other qubit j is $+1$ for $j < k$, and -1 for $j \geq k$ due to a strong linear term pinning the vertical j bits. In this case, every QUBO quadratic contribution to the left of k is negative while all to the right are positive, and the total contribution to the QUBO energy is small,



with a weight of $w = w_i - \sum_{j<k} |w_{ij}| + \sum_{j>k} |w_{ij}|$. However, if one adds a single defect to bit i at k, it looks like a $+1$ state to the left and -1 to the right, satisfying every QUBO quadratic term at the cost of adding one defect worth of energy,



with a weight of $w' = w_i + \sum_j |w_{ij}| - \delta$. To guarantee the MWIS encodes a valid solution, the weight of the zero-defect solution must be larger $w \geq w'$. Given a QUBO contribution will always contribute a positive weight ($w > w_i$), this guarantee is equivalent to enforcing that the defect cost is greater than the sum on all w for each bit

$$\delta > \max_i \sum_j |w_{ij}|. \tag{B3}$$

If both the linear and quadratic constraints are satisfied by normalizing the weights correctly, the MWIS is guaranteed to be a zero-defect state and thus encode the QUBO solution.

APPENDIX C: RESTRICTED QUBO CONNECTIVITY

While it is useful to envision arbitrary-connectivity QUBO problems, its application comes at the practical cost of encoding overhead. For example, encoding a 5-bit problem takes around a 16×16 lattice, which is on the upper limit of today's Rydberg atom array system [10]. A more near-term solution is to specify graphs with a constrained connectivity that naturally fit more bits onto today's hardware. A natural restriction is a nearest-neighbor 2D connectivity, such as an example graph shown in Fig. 11(a). Just like any arbitrary QUBO problem can be mapped to a UDG-MWIS problem with a quadratic overhead, a restricted 2D QUBO problem can be mapped onto UDG-MWIS with only a constant overhead.

In order to construct particular restricted connectivity QUBO problems for the particular topology of Fig. 11, it is instructive to introduce three new gadgets, which are extensions of the crossing gadget. The first gadget is a square clique of four vertices, which is similar to the four-vertex clique of the QUBO gadget, shown in Fig. 11(b). The four MWIS of the clique represent four possible states of two qubits; for this mapping, we choose the top right vertex to be the $++$ state, and the bottom right vertex to be the $+-$ state, etc. In order to encode a QUBO interaction between the bits, we likewise bias the weights of each vertex as shown in Fig. 11(b).

An additional gadget can encode ferromagnetic ($J > 0$) or antiferromagnetic ($J < 0$) interactions between adjacent bits, as shown in Fig. 11(c). The independent set restriction naturally encodes nn-type interactions, while QUBO usually requires ZZ-type interactions, which can be converted back and forth using linear terms. In this way, the interaction between adjacent bits can be encoded by adding one ($J < 0$) or two ($J > 0$) ancilla vertices

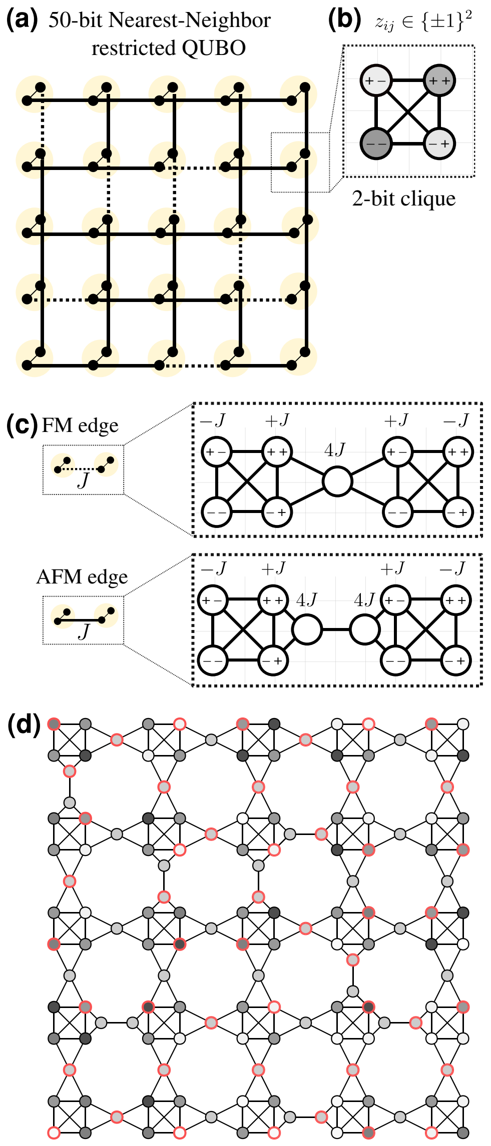


FIG. 11. An example 2D restricted-connectivity graph and reduction to UDG-MWIS. (a) A particular topology of bits (vertices) and quadratic terms (edges). The original graph can be mapped into a UDG-MWIS using some gadgets. (b) Two bits can be represented by a clique of four vertices, with each state ($\pm\pm$) represented by a single vertex of the clique. Linear and quadratic interactions are represented by biasing the weights of the clique. (c) Interactions between neighbors can be represented by adding ancilla vertices. (d) The original restricted-connectivity QUBO problem as mapped to a UDG-MWIS problem, where the ground state encodes the solution to the QUBO problem. Here, quadratic QUBO weights are chosen to be $\pm J$. This example graph has 50 bits in the original graph and an extent of 13×13 in the mapped UDG graph, which naturally fits onto today's Rydberg atom array hardware.

in between each clique within the unit-disk radius. Ultimately, the absolute value of the interaction is encoded into the weight of these interaction vertices.

For an example of how this interaction gadget works, consider the one vertex antiferromagnetic interaction and gadget of Fig. 11(c). If the horizontal bit of the left clique is in the $+1$ state (e.g., on the right side of the clique), the ancilla vertex is blockaded from being part of the independent set, and likewise if the horizontal bit of the right clique is in the -1 state. Thus, the effective interaction, encoded into the condition of the ancilla vertex of weight w_{ij} being included in the independent set, is $w_{ij}(1-z_i)(1+z_j)/4$. Similarly, the two-vertex ferromagnetic interaction is encoded into the weight as $w_{ij}(1-(1-z_i)(1+z_j)/4)$, as the ancilla vertex is only blockaded for one configuration instead of three. Note that the antiferromagnetic gadget has a negative sign in front of the zz term, as required, and similar for the ferromagnetic gadget. After correcting the linear offsets, the biases for each vertex of the gadget are shown in Fig. 11(c), with $w_{ij}=J$.

Finally, one must guarantee that the ground state does in fact map to the codespace of valid solutions, with proper normalization as described in Appendix B. Here, a solution is valid if each four-vertex clique has at least one vertex in the maximum independent set. This may be guaranteed by increasing the zero-bias weight of the four-vertex clique to be much larger than any other scale. One guaranteed offset is $U=8J$, where J is the largest coupling strength between adjacent bit cliques. This condition is set because, if the weight is smaller, an independent set vertex in the clique can be replaced with two adjacent vertices of the interaction gadget, which each have a weight of $4|J|$. Thus, this bias guarantees the correct ground state. An example set of weights, which encodes a graph with random bonds $\pm J$, is shown in Fig. 11(d).

It should be emphasized that Fig. 11 is just one *particular* example of a local connectivity encoding for unit-disk graphs. In practice, there may be many different encodings of many different restricted-connectivity graphs. Due to the nature of Rydberg atom arrays, which reconstruct the graph for each shot, these neutral-atom platforms are much more flexible in the connectivities of the problems they solve, potentially even on a shot-by-shot basis. This is in contrast with other architectures such as superconducting qubits, which have a fixed qubit connectivity and require a lengthy fabrication process to modify their topology.

Additionally, these local-connectivity graphs can encode more nonlocal problems, by increasing the ferromagnetic weight of edges such that the ground state of adjacent vertices are always the same. In this way, choosing a large ferromagnetic weight recreates the copy gadget and, by extension, may recreate the crossing lattice of the all-to-all QUBO problem shown in Fig. 7. Furthermore, such a local-connectivity graph may recreate other hardware's configuration. For instance, the DWAVE Chimera graph [46] consists of sets of eight bipartite connected bits in a unit cell, which are connected colinearly with adjacent unit cells. The same connectivity can be reproduced by choosing some large ferromagnetic J terms appropriately on the grid of Fig. 11. It should be emphasized that due to the reconfigurable nature of Rydberg atom arrays, it is trivial to modify the topology of the connectivity. For example, on one shot, a Rydberg atom array system could recreate the DWAVE Chimera topology, while, on the next shot, it may recreate the DWAVE Pegasus topology, and so forth.

APPENDIX D: FACTORING GADGET

In this Appendix, we elaborate on the factoring gadget introduced in Fig. 8. The factoring gadget is designed such that the MWIS space corresponds to the satisfying assignments

$$s_{i,j}+2c_{i,j}=p_{i,j}q_{i,j}+s_{i+1,j-1}+c_{i-1,j}, \tag{D1}$$
$$q_{i+1,j}=q_{i,j}, \tag{D2}$$
$$p_{i,j+1}=p_{i,j}. \tag{D3}$$

We first focus on the constraint (D1) since the other two constraints are easy to satisfy with a combination of copy and crossing gadgets. To simplify notations, let us rewrite the constraint (D1) as $ab+c+d=2e+f$ between binary variables $a,b,c,d,e,f\in\{0,1\}$. We further rewrite this constraint as a conjunction of two simple constraints, namely

$$z=ab, \tag{D4}$$
$$z+c+d=2e+f. \tag{D5}$$

We obtain a MWIS representation of the first constraint directly from the crossing gadget [see Fig. 12(a)]. As already discussed in the main text, the interior vertices of the crossing gadget encode the information about the variables on the boundary. Specifically, the lower left interior vertex (representing z) is in the MWIS if and only if both the top and the right outer vertices (representing a and b, respectively) are also in the MWIS, thus exactly representing $ab=z$.

The MWIS representation of the second constraint is given in Fig. 12(b). One can check by exhaustive search that the MWISs of this gadget indeed represent exactly all satisfying assignments of $z+c+d=2e+f$. To obtain the MWIS representation of $ab+c+d=2e+f$, we thus simply join the graphs in Figs. 12(a) and 12(b) at the common vertex z. Note that the total weight of the vertex z in this joint graph is the sum of its weights in each individual graph. One can easily identify this joint structure in the full factoring gadget given in Fig. 8(b). The remaining parts of this gadget are simply formed by combining it with copy and crossing gadgets that satisfy (D2) and (D3) and to route

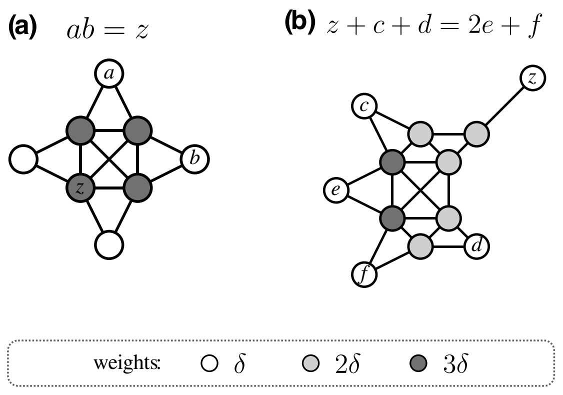


FIG. 12. The two basic components of the factoring gadget. The gadget in (a) is a crossing gadget. Besides its use in routing effective variables a and b, it also serves as a tool to access the value of the product of the variables, $z = ab$, at the indicated interior vertex. The gadget in (b) is the MWIS representation of the constraint $z + c + d = 2e + f$.

the variables to positions where they can be accessed also by neighboring factoring gadgets.

[1] A. Schrijver *et al.*, *Combinatorial Optimization: Polyhedra and Efficiency* (Springer, Berlin, 2003), Vol. 24.
[2] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms* (Springer, Heidelberg, 2008).
[3] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum Computation by Adiabatic Evolution, arXiv:0001106 (2000).
[4] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem, Science **292**, 472 (2001).
[5] T. Kadowaki and H. Nishimori, Quantum annealing in the transverse Ising model, Phys. Rev. E **58**, 5355 (1998).
[6] A. Das and B. K. Chakrabarti, Colloquium: Quantum annealing and analog quantum computation, Rev. Mod. Phys. **80**, 1061 (2008).
[7] T. Albash and D. A. Lidar, Adiabatic quantum computation, Rev. Mod. Phys. **90**, 015002 (2018).
[8] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028 (2014).
[9] A. Lucas, Ising formulations of many np problems, Front. Phys. **2**, 5 (2014).
[10] S. Ebadi *et al.*, Quantum optimization of maximum independent set using Rydberg atom arrays, Science **376**, 1209 (2022).
[11] D. Jaksch, J. I. Cirac, P. Zoller, S. L. Rolston, R. Côté, and M. D. Lukin, Fast Quantum Gates for Neutral Atoms, Phys. Rev. Lett. **85**, 2208 (2000).
[12] M. Saffman, T. G. Walker, and K. Mølmer, Quantum information with Rydberg atoms, Rev. Mod. Phys. **82**, 2313 (2010).
[13] 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 (2018).
[14] M. Kim, K. Kim, J. Hwang, E.-G. Moon, and J. Ahn, Rydberg quantum wires for maximum independent set problems, Nat. Phys. **18**, 755 (2022).
[15] 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 (2022).
[16] A. Byun, M. Kim, and J. Ahn, Finding the Maximum Independent Sets of Platonic Graphs using Rydberg Atoms, PRX Quantum **3**, 030305 (2022).
[17] W. Lechner, P. Hauke, and P. Zoller, A quantum annealing architecture with all-to-all connectivity from local interactions, Sci. Adv. **1**, e1500838 (2015).
[18] X. Qiu, P. Zoller, and X. Li, Programmable Quantum Annealing Architectures with Ising Quantum Wires, PRX Quantum **1**, 020311 (2020).
[19] C. Dlaska, K. Ender, G. B. Mbeng, A. Kruckenhauser, W. Lechner, and R. van Bijnen, Quantum Optimization via Four-Body Rydberg Gates, Phys. Rev. Lett. **128**, 120503 (2022).
[20] B. Clark, C. Colbourn, and D. Johnson, Unit disk graphs, Discrete Math. **86**, 165 (1990).
[21] We also show that circuit satisfiability problems can be mapped into UDG-MWIS and so all other NP problems can be mapped through circuit satisfiability if no better low-overhead mapping is found.
[22] J.-G. Liu *et al.*, (to be published 2022).
[23] H. Weimer, M. Müller, I. Lesanovsky, P. Zoller, and H. P. Büchler, A Rydberg quantum simulator, Nat. Phys. **6**, 382 (2010).
[24] A. Browaeys and T. Lahaye, Many-body physics with individually controlled Rydberg atoms, Nat. Phys. **16**, 132 (2020).
[25] A. M. Kaufman and K.-K. Ni, Quantum science with optical tweezer arrays of ultracold atoms and molecules, Nat. Phys. **17**, 1324 (2021).
[26] S. Ebadi, T. T. Wang, H. Levine, A. Keesling, G. Semeghini, A. Omran, D. Bluvstein, R. Samajdar, H. Pichler, W. W. Ho, S. Choi, S. Sachdev, M. Greiner, V. Vuletić, and M. D. Lukin, Quantum phases of matter on a 256-atom programmable quantum simulator, Nature **595**, 227 (2021).
[27] P. Scholl, M. Schuler, H. J. Williams, A. A. Eberharter, D. Barredo, K.-N. Schymik, V. Lienhard, L.-P. Henry, T. C. Lang, T. Lahaye, A. M. Läuchli, and A. Browaeys, Quantum simulation of 2D antiferromagnets with hundreds of Rydberg atoms, Nature **595**, 233 (2021).
[28] T. M. Graham *et al.*, Multi-qubit entanglement and algorithms on a neutral-atom quantum computer, Nature **604**, 457 (2022).
[29] I. S. Madjarov, J. P. Covey, A. L. Shaw, J. Choi, A. Kale, A. Cooper, H. Pichler, V. Schkolnik, J. R. Williams, and M. Endres, High-fidelity entanglement and detection of alkaline-earth Rydberg atoms, Nat. Phys. **16**, 857 (2020).
[30] A. W. Young, W. J. Eckner, W. R. Milner, D. Kedar, M. A. Norcia, E. Oelker, N. Schine, J. Ye, and A. M. Kaufman, Half-minute-scale atomic coherence and high relative stability in a tweezer clock, Nature **588**, 408 (2020).

[31] K. Singh, S. Anand, A. Pocklington, J. T. Kemp, and H. Bernien, Dual-Element, Two-Dimensional Atom Array with Continuous-Mode Operation, Phys. Rev. X **12**, 011040 (2022).

[32] J. T. Zhang, L. R. B. Picard, W. B. Cairncross, K. Wang, Y. Yu, F. Fang, and K.-K. Ni, An optical tweezer array of ground-state polar molecules, Quantum Sci. Technol. **7**, 035006 (2022).

[33] L.-M. Steinert, P. Osterholz, R. Eberhard, L. Festa, N. Lorenz, Z. Chen, A. Trautmann, and C. Gross, Spatially programmable spin interactions in neutral atom arrays, arXiv:2206.12385 (2022).

[34] A. Omran, H. Levine, A. Keesling, G. Semeghini, T. T. Wang, S. Ebadi, H. Bernien, A. S. Zibrov, H. Pichler, S. Choi, J. Cui, M. Rossignolo, P. Rembold, S. Montangero, T. Calarco, M. Endres, M. Greiner, V. Vuletić, and M. D. Lukin, Generation and manipulation of Schrödinger cat states in Rydberg atom arrays, Science **365**, 570 (2019).

[35] Note that the particular scale of U_{uv} does not change the low-energy properties (i.e., the energy of the states corresponding to independent sets) as long as $U_{uv} > \max_w \delta_w$.

[36] H. Pichler, S.-T. Wang, L. Zhou, S. Choi, and M. D. Lukin, Computational complexity of the Rydberg blockade in two dimensions, arXiv:1809.04954 (2018).

[37] Note that, in general, this strategy could result in edges appearing in multiple constraints, producing inhomogeneous edge weights. However, because the satisfying assignments are independent sets, we can always homogenize the edge constraint by considering an equivalent problem of homogeneous edge weights by choosing a large enough $U \gg \delta$.

[38] C. Moore and S. Mertens, *The Nature of Computation* (Oxford University Press, Oxford, 2011).

[39] V. Choi, Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design, Quantum Inf. Process. **10**, 343 (2010).

[40] S. Knysh and V. N. Smelyanskiy, Adiabatic Quantum Computing in systems with constant inter-qubit couplings, arXiv:0511131 (2005).

[41] Any weight larger than 2δ for the ancillary vertices would work. The choice 4δ is convenient, since it homogenized defect energies when crossing gadgets are combined with copy gadgets.

[42] C. J. Burges, Factoring as optimization, Microsoft Research MSR-TR-200 (2002).

[43] R. Dridi and H. Alghassi, Prime factorization using quantum annealing and computational algebraic geometry, Sci. Rep. **7**, 43048 (2017).

[44] J.-G. Liu, X. Gao, M. Cain, M. D. Lukin, and S.-T. Wang, Computing solution space properties of combinatorial optimization problems via generic tensor networks, arXiv:2205.03718 (2022).

[45] D. Coudert, D. Mazauric, and N. Nisse, in *International Symposium on Experimental Algorithms* (Springer, Berlin, 2014), p. 46.

[46] D-wave QPU architecture: Topologies. https://docs.dwavesys.com/docs/latest/c˙gs˙4.html.