COMPUTING SOLUTION SPACE PROPERTIES OF COMBINATORIAL OPTIMIZATION PROBLEMS VIA GENERIC TENSOR NETWORKS

JIN-GUO LIU*, XUN GAO†, MADELYN CAIN†, MIKHAIL D. LUKIN†, AND SHENG-TAO WANG‡

Abstract. We introduce a unified framework to compute the solution space properties of a broad class of combinatorial optimization problems. These properties include finding one of the optimum solutions, counting the number of solutions of a given size, and enumeration and sampling of solutions of a given size. Using the independent set problem as an example, we show how all these solution space properties can be computed in the unified approach of generic tensor networks. We demonstrate the versatility of this computational tool by applying it to several examples, including computing the entropy constant for hardcore lattice gases, studying the overlap gap properties, and analyzing the performance of quantum and classical algorithms for finding maximum independent sets.

Key words. solution space property, tensor networks, maximum independent set, independence polynomial, generic programming, combinatorial optimization

AMS subject classifications. 15A69, 05C31, 14N10

1. Introduction. An important class of problems in graph theory and combinatorial optimization can be formulated as satisfiability problems involving constraints specified over a vertex and its neighborhood. These problems include, for example, the independent set problem, the cutting problem, dominating set, set packing, set covering, vertex coloring, K-SAT, the clique problem, and the vertex cover problem [45]. These problems have a wide range of applications in scheduling, logistics, wireless networks and telecommunication, and computer vision, among others [13, 55]. Finding an optimum solution for these problems is typically NP-hard in the worst case [34].

In this Article, we introduce a unified framework to compute a broad class of properties associated with the solutions of these problems, beyond just finding an optimum solution. We call them *solution space properties*. In practice, these can be much harder to compute (corresponding e.g. to #P-complete class [45]). However, these properties can be crucial for understanding detailed properties of hard combinatorial optimization problems. For example, for the independent set problem, these *solution space properties* can include not only the maximum or minimum independent set size but also the number of sets at a given size, enumeration of all sets at a given size, and direct sampling of such sets when they are too large to be fit into memory. They can be used to understand the hardness of finding an optimum solution for a given problem instance and the performance of a specific solver. For example, the number of configurations at different sizes can inform how likely a simulated annealing algorithm will be trapped in local minima at certain sizes [56]. The pair-wise Hamming distance distribution of configurations at a given size can indicate the presence or absence of the overlap gap property [26, 25], which can be used to bound the performance of local optimization algorithms. In a recent experiment based on a Rydberg atom array quantum computer, the counting and the configuration space connectivity information was used to find maximum independent set (MIS) problem instances that are hard for simulated annealing and to evaluate the corresponding quantum algorithm performance [19]. The need to understand these important aspects of combinatorial optimization motivates us to find methodologies to compute these solution space properties.

*Department of Physics, Harvard University, Cambridge, Massachusetts 02138, USA; QuEra Computing Inc., 1284 Soldiers Field Road, Boston, MA, 02135, USA (jinguoliu@g.harvard.edu).

†Department of Physics, Harvard University, Cambridge, Massachusetts 02138, USA (xungao@g.harvard.edu, contributed equally with Jin-Guo Liu to this work.).

‡QuEra Computing Inc., 1284 Soldiers Field Road, Boston, MA, 02135, USA

To this end, we show how to obtain all of these seemingly unrelated properties in a unified approach using *generic tensor networks*. Tensor networks are a computational model widely used in condensed matter physics [46], quantum computing [44], big data [15], mathematics [47] and combinatorial optimization [9, 8, 38]. They are also known as the sum-product networks in probabilistic modeling [10] or `einsum` in linear algebra libraries such as NumPy [32]. Recent progress in simulating quantum circuits with tensor networks [31, 48, 36] makes it possible to contract a randomly structured sparse tensor network with up to thousands of tensors in a reasonable time. In previous studies, the data types of the tensor elements are typically restricted to standard number types such as real numbers and complex numbers. Here, we extend to *generic tensor networks* by generalizing the tensor element data types to any type that has the algebraic structure of a commutative semiring. In what follows, for clarity of presentation, we focus on the independent set problem in the main text, and show how to compute the solution space properties for other combinatorial optimization problems in Appendix B. The latter includes cutting, matching, vertex coloring, satisfiability, dominating set, set packing, set covering, and the clique problem.

The paper is organized as follows. We first introduce the basic concepts of tensor networks and generic programming in Section 2 and Section 3. Then we show how to reduce the independent set problem to a tensor network contraction problem in Section 4. Subsequently, we explain how to engineer the element types to compute various solution space properties in Section 6, Section 7, and Section 8. Lastly, we provide three example applications in Section 9 to demonstrate the versatility of our tool. A benchmark to demonstrate the performance of our algorithms can be found in both the Section SM1 and the code repository [1].

2. Tensor networks. A tensor network is a multi-linear map from a collection of labelled tensors $\mathcal{T}$ to an output tensor. It is formally defined as follows.

DEFINITION 2.1 (Tensor Network [16, 46]). *A tensor network is a multi-linear map specified by a triple of $\mathcal{N} = (\Lambda, \mathcal{T}, \sigma_o)$, where Λ is a set of symbols (or labels), $\mathcal{T} = \{T^{(1)}_{\sigma_1}, T^{(2)}_{\sigma_2}, \ldots, T^{(M)}_{\sigma_M}\}$ is a set of tensors as the inputs, and σ_o is a string of symbols labelling the output tensor. Each $T^{(k)}_{\sigma_k} \in \mathcal{T}$ is labelled by a string $\sigma_k \in \Lambda^{r(T^{(k)})}$, where $r\left(T^{(k)}\right)$ is the rank of $T^{(k)}$. The multi-linear map or the **contraction** on this triple is*

$$(2.1) \qquad O_{\sigma_o} = \sum_{\Lambda \backslash \sigma_o} \prod_{k=1}^{M} T^{(k)}_{\sigma_k},$$

where the summation runs over all possible configurations over the set of symbols absent in the output tensor.

For example, the matrix multiplication can be specified as a tensor network

$$(2.2) \qquad \mathcal{N}_{\text{matmul}} = \left(\{i, j, k\}, \{A_{ij}, B_{jk}\}, ik\right),$$

where A_{ij} and B_{jk} are input matrices (two-dimensional tensors), and (i, k) are labels associated to the output. The contraction is defined as $O_{ik} = \sum_j A_{ij} B_{jk}$, where the subscripts are for tensor indexing, and the tensor dimensions with the same label must have the same size. The graphical representation of a tensor network is an open hypergraph that having open hyperedges, where an input tensor is mapped to a vertex and a label is mapped to a hyperedge that can connect an arbitrary number of vertices, while the labels appearing in the output tensor are mapped to open hyperedges. Our notation is a minor generalization of the standard tensor network notation used in physics as we do not restrict the number of times a label

2

87 can appear in the tensors to two. While this generalized form is equivalent in representation
88 power, it can have smaller contraction complexity as will be illustrated in Section SM2.

89 **Example 1.**

$$\Lambda = \{i, j, k, l, m\},$$
$$\mathcal{T} = \{A_{jkm}, B_{mil}, V_{jm}\},$$
$$\sigma_o = ijk,$$

(2.3)

91 is a tensor network that can be evaluated as $O_{ijk} = \sum_{ml} A_{jkm} B_{mil} V_{jm}$. Its hypergraph represen-
92 tation is shown below, where we use different colors to represent different hyperedges.
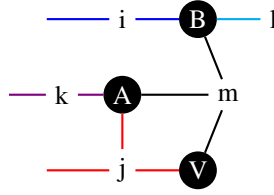
93 **3. Generic programming tensor contractions.** In previous works relating tensor
94 networks and combinatorial optimization problems [38, 8], the element types in the tensor
95 networks are limited to standard number types such as floating-point numbers and integers.
96 We propose to use more general element types with a certain algebraic property. With
97 different data types, we can solve different problems within the same unified framework.
98 This idea of using the same program for different purposes is also called generic
99 programming in computer science:

100 DEFINITION 3.1 (Generic programming [54]). *Generic programming is an approach to*
101 *programming that focuses on designing algorithms and data structures so that they work in*
102 *the most general setting without loss of efficiency.*

103 This definition of generic programming covers two major aspects: "work in the most general
104 setting" and "without loss of efficiency". By the most general setting, we mean that a single
105 program should work correctly for the most general input data types. For example, suppose
106 we want to write a function that raises an element to a power, $f(x, n) := x^n$. One can easily
107 write a function for standard number types that computes the power of x in $O(\log(n))$ steps
108 using the multiply and square trick. Generic programming does not require x to be a standard
109 number type; instead, it treats x as an element with an associative multiplication operation $\odot$
110 and a multiplicative identity $\mathbb{1}$. Then, when the program takes a matrix as an input instead of a
111 standard number type, it computes the matrix power correctly without rewriting the program.
112 The second aspect is about efficiency. For dynamically typed languages such as Python,
113 the type information is not available for type-specific optimizations at the compilation stage.
114 Therefore, one can easily write code that works for general input types, but the efficiency
115 is not guaranteed; for example, the speed of computing the matrix multiplication between
116 two NumPy arrays with Python objects as elements is much slower than statically typed (i.e.
117 the type information can be accessed at the compilation stage) languages such as C++ and
118 Julia [7]. C++ uses templates for generic programming while Julia takes advantage of just-
119 in-time compilation and multiple dispatches. When these languages "see" a new input type,
120 the compiler recompiles the generic program for the new type to generate an efficient binary.
121 A myriad of optimizations can be done during the compilation. For example, the compiler
122 can optimize the memory layout of immutable elements with fixed sizes in an array to speed
123 up array indexing. In Julia, if a type is immutable and contains no references to other values,

3

124 an array of that type can even be compiled to graphics processing units (GPU) for faster
125 computation [6].
126   This motivates us to identify the most general tensor element type allowed in a tensor
127 network contraction program. We find that as long as the tensor elements are members of
128 a commutative semiring, the tensor network contraction will be well defined and the result
129 will be independent of the contraction order. In contrast with a field, a commutative semiring
130 does not need not to have an additive inverse and a multiplicative inverse. Giving up these
131 nice properties of fields has significant implications for tensor computation: tensor network
132 compression algorithms might not be applicable because matrix factorization is NP-hard for
133 commutative semirings [53] and matrix multiplication faster than $O(n^3)$ does not exist for
134 an algebra without an additive inverse [37]. Here, we only use the commutative properties
135 of an algebra for optimizing the tensor network contraction order. To define a commutative
136 semiring with the addition operation $\oplus$[1] and the multiplication operation $\odot$ on a set S, the
137 following relations must hold for any arbitrary three elements $a, b, c \in S$.

138 $$(a \oplus b) \oplus c = a \oplus (b \oplus c) \qquad \rhd \text{ commutative monoid } \oplus \text{ with identity } \mathbb{0}$$

139 $$a \oplus \mathbb{0} = \mathbb{0} \oplus a = a$$

140 $$a \oplus b = b \oplus a$$

142 $$(a \odot b) \odot c = a \odot (b \odot c) \qquad \rhd \text{ commutative monoid } \odot \text{ with identity } \mathbb{1}$$

143 $$a \odot \mathbb{1} = \mathbb{1} \odot a = a$$

144 $$a \odot b = b \odot a$$

146 $$a \odot (b \oplus c) = a \odot b \oplus a \odot c \qquad \rhd \text{ left and right distributive}$$

147 $$(a \oplus b) \odot c = a \odot c \oplus b \odot c$$

149/150 $$a \odot \mathbb{0} = \mathbb{0} \odot a = \mathbb{0}$$

151 In the following sections, we will show how to compute solution space properties of
152 independent sets using the same tensor network contraction algorithm by engineering tensor
153 element algebra. The Venn diagram in Figure 1 shows the different types of algebra we will
154 introduce in the main text and their relation, and Table 1 summarizes which solution space
155 properties can be computed by which tensor element types.

156   **4. Tensor network representation of independent sets.** This section describes the
157 reduction of the independent set problem to a tensor network contraction problem. An
158 alternative interpretation, perhaps more accessible to physicists, can be found in
159 Appendix A, where we introduce the reduction from the energy model of hardcore lattice
160 gases [18, 21]. Let $G = (V, E)$ be an undirected graph with each vertex $v \in V$ being
161 associated with a integer weight w_v. An independent set $I \subseteq V$ is a set of vertices that for any
162 vertex pair $u, v \in I$, we have $(u, v) \notin E$; we refer to this constraint as the independence
163 constraint. The independent set problem on G can be encoded as a tensor network $\mathcal{N}_{IS}(G)$

164 (4.1)
$$\Lambda = \{s_v \mid v \in V\},$$
$$\mathcal{T} = \{W^{(v)}_{s_v} \mid v \in V\} \cup \{B^{(u,v)}_{s_u s_v} \mid (u, v) \in E\},$$
$$\sigma_o = \varepsilon,$$

---

[1] We use the $\oplus$ operator throughout this paper to denote the generic addition, which is not the logical XOR
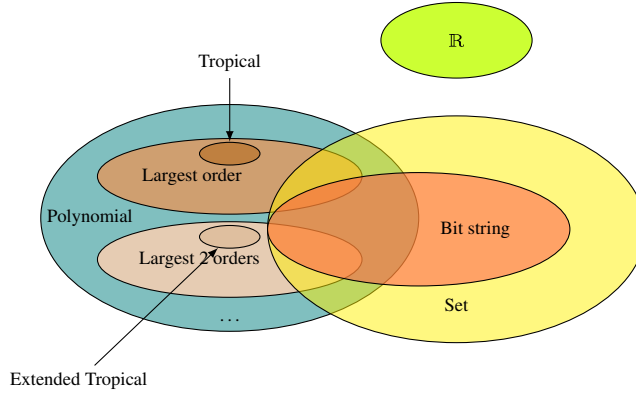operation, in which the symbol typically represents in computer science.

Figure 1: The tensor network element types used in this work and their relations. The overlap between two ellipses indicates that a new algebra can be created by combining those two types of algebra. "Largest order" and "Largest 2 orders" mean truncating the polynomial by only keeping its largest or largest two orders. The purposes of these element types can be found in Table 1.

Element type	Solution space property
$\mathbb{R}$	Counting of all independent sets
Polynomial (Equation (5.2: PN))	Independence polynomial
Tropical (Equation (6.4: T))	Maximum independent set size
Extended tropical of order k (Equation (8.1: Tk))	Largest k independent set sizes
Polynomial truncated to k-th order (Equation (6.3: P1) and Equation (6.6: P2))	k largest independent sizes and their degeneracy
Set (Equation (7.1: SN))	Enumeration of independent sets
Sum-Product expression tree (Equation (7.7: EXPR))	Sampling of independent sets
Polynomial truncated to largest order combined with bit string (Equation (7.5: S1))	Maximum independent set size and one of such configurations
Polynomial truncated to k-th order combined with set (Equation (7.3: P1+SN))	k largest independent set sizes and their enumeration

Table 1: Tensor element types and the independent set properties that can be computed using them.

165   where for each vertex v, we define a parameterized rank-one tensor associated with it as

166   (4.2)
$$W^{(v)} = \begin{pmatrix} 1 \\ x_v^{w_v} \end{pmatrix},$$

5

and for each edge $(u, v) \in E$, we define a matrix B as

$$B^{(u,v)} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}. \tag{4.3}$$

We map each vertex $v \in V$ to a label $s_v \in \{0, 1\}$, where we use 0 or 1 to denote a vertex is absent or present in I, respectively. These labels can be used as subscripts of tensors to index tensor elements, e.g. $W_0^{(v)} = 1$ is the first element associated with $s_v = 0$ and $W_1^{(v)} = x_v^{w_v}$ is the second element associated with $s_v = 1$, where x_v is an element of some commutative semiring (e.g., listed in Table 1) associated with vertex v and its power with an integer is defined by repeated multiplication. The labels associated to the output tensor is an empty string ε, meaning this tensor has rank 0, i.e. the output is a scalar. The independence constraint is encoded in the edge tensors, where we use $B_{11}^{(u,v)} = 0$ to denote two vertices connected by an edge (u, v) cannot be both in the independent set. The contraction of this tensor network is

$$P(G) = \sum_{s_1, s_2, \ldots, s_{|V|} = 0}^{1} \prod_{v \in V} W_{s_v}^{(v)} \prod_{(u,v) \in E} B_{s_u s_v}^{(u,v)}, \tag{4.4}$$

where the summation runs over all $2^{|V|}$ vertex configurations $\{s_1, s_2, \ldots, s_{|V|}\}$ and accumulates the product of tensor elements to the output P. A vertex tensor element $W_{s_v}^{(v)}$ contributes a multiplicative factor $x_v^{w_v}$ whenever v is in the set.

Example 2. Here, we show a minimum example of mapping the independent problem of a 2-vertex complete graph K2 (left) to a tensor network (right).
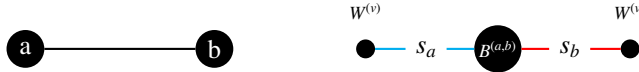
In the graphical representation of the tensor network on the right panel, we use a circle to represent a tensor, a cyan-colored hyperedge to represent the degree of freedom s_a, and a hyperedge in red to represent the degree of freedom s_b. Tensors sharing the same degree of freedom are connected by the same hyperedge. The contraction of this tensor network has the following form:

$$P(\text{K2}) = \begin{pmatrix} 1 & x_a^{w_a} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ x_b^{w_b} \end{pmatrix}$$
$$= 1 + x_a^{w_a} + x_b^{w_b}. \tag{4.5}$$

The resulting polynomial represents 3 different independent sets {}, {a}, and {b} with weights 0, w_a, and w_b, respectively.

For a general graph, it is computationally inefficient to evaluate Equation (4.4) by directly summing up the $2^{|V|}$ products. A better approach to evaluate a tensor network is: to find a good pair-wise tensor contraction order as a binary tree and then contract two tensors at a time by this order.

THEOREM 4.1. *The tensor network in Equation (4.1) for the independent set problem on graph $G = (V, E)$ can be contracted in $cc(\mathcal{N}_{\text{IS}}(G)) = O(|E|)2^{O(\text{tw}(G))}$ number of additions and multiplications.*

Proof. Let us denote the line graph, a graph obtained by mapping an edge in the original graph to a vertex and connect two vertices if and only if their associated edges in the original graph share a common vertex, of the hypergraph representation of a tensor network $\mathcal{N}$ as $L(\mathcal{N})$. A contraction order of $\mathcal{N}$ corresponds to a tree decomposition of $L(\mathcal{N})$ and the

6

203 largest intermediate tensor has a rank equal to the width of its tree decomposition [44].
204 Therefore, an optimal (in terms of space complexity) contraction order corresponds to the
205 tree decomposition of $L(\mathcal{N})$ with the smallest width (or the treewidth tw($L(\mathcal{N})$). The
206 contraction complexity is $O(|\mathcal{N}|)2^{O(\text{tw}(L(\mathcal{N})))}$, where $|\mathcal{N}|$ is the number of tensors in $\mathcal{N}$ [44].
207 For the independent set problem on graph $G = (V, E)$, the line graph of the hypergraph
208 representation of the tensor network in Equation (4.1) is isomorphic to the graph G up to
209 some isolated vertices, hence the contraction complexity of this tensor network with an
210 optimal contraction order is $O(|E|)2^{O(\text{tw}(G))}$. ꛦ

211   In practice, it is difficult to find an optimal contraction order for large tensor networks
212 because finding the treewidth is a well-known NP-hard problem. However, it is easy to find
213 a close-to-optimal contraction order within typically a few minutes using a heuristic
214 algorithm [38, 36]. For large-scale applications, it is also possible to slice over certain
215 degrees of freedom to reduce the space complexity, i.e. loop over possible combinations of
216 certain degrees of freedom so that one can have a smaller tensor network inside the loop
217 since these degrees of freedoms are fixed.

218 **Example 3.** In this example, we map a 5-vertex graph (left) to a tensor network (right) and
219 show how optimizing the contraction order reduces the time and space complexities.
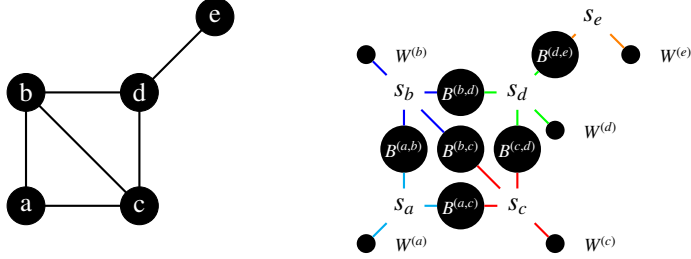
220   One can represent a possible pair-wise contraction of tensors as a binary tree structure:
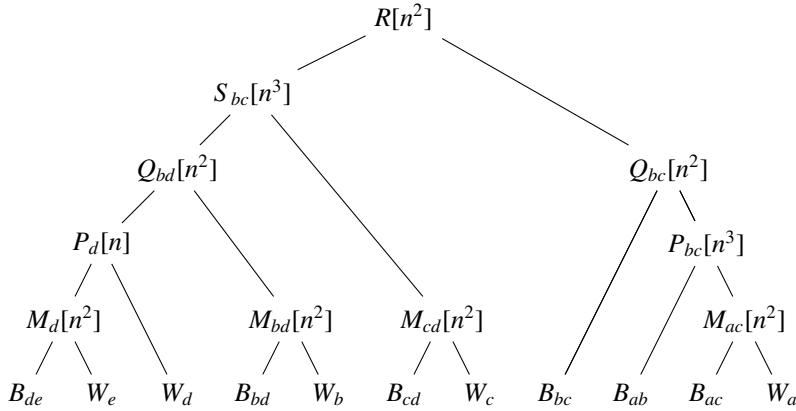
221   The contraction process goes from bottom to top, where the root node stores the
222 contraction result, the leaves are input tensors, and the rest of the nodes are all intermediate
223 contraction results. Tensor subscripts are indices so that the number of subscripts indicates
224 the space complexity to store this tensor. The contraction complexity to generate a tensor is
225 annotated in the square brackets, where n is the dimension of the degree of freedoms, which
226 is 2 in a tensor network mapped from an independent set problem. One can easily check the
227 largest tensor in contraction has a space complexity $O(n^2)$ and this is the smallest among all
228 possible contraction trees, i.e. the treewidth of the original 5-vertex graph is 2. The time
229 complexity is $O(n^3)$, which is much smaller than that of direct evaluation $O(n^5)$.

7

5. Independence polynomial. Let $x_i = x$ and $w_i = 1$, Equation (4.4) corresponds to the independence polynomial:

$$(5.1) \qquad I(G, x) = \sum_{k=0}^{\alpha(G)} a_k x^k,$$

where a_k is the number of independent sets of size k, $\alpha(G) \equiv \max_I \sum_{v \in I} w_v$ is the size of a maximum independent set and it is also called the independence number. An independence polynomial is a useful graph characteristic related to, for example, the partition functions [39, 57] and Euler characteristics of the independence complex [11, 40]. By assigning a real number to x, one can evaluate this independence polynomial for this specific value directly using tensor network contraction. For example, the total number of independent sets can be evaluated as $I(G, 1)$. However, instead of evaluating this polynomial for a certain value, we are more interested in knowing the coefficients of this polynomial, because this quantity tells us the counting of independent sets at different sizes. To this end, let us create a polynomial number data type by representing a polynomial $a_0 + a_1 x + \ldots + a_k x^k$ as a coefficient vector $a = (a_0, a_1, \ldots, a_k) \in \mathbb{R}^k$, e.g. x is represented as $(0, 1)$. Then we can define an algebra among coefficient vectors, including a redefinition of additive identity and multiplicative identity. To avoid potential confusion, let us denote the additive identity as $\mathbb{0}$, and the multiplicative identity is $\mathbb{1}$. The algebra between the polynomials number a of order k_a and b of order k_b is specified as

$$(5.2: \text{PN}) \qquad \begin{aligned} a \oplus b &= (a_0 + b_0, a_1 + b_1, \ldots, a_{\max(k_a,k_b)} + b_{\max(k_a,k_b)}), \\ a \odot b &= (a_0 + b_0, a_1 b_0 + a_0 b_1, a_2 b_0 + a_1 b_1 + a_0 b_2, \ldots, a_{k_a} b_{k_b}), \\ \mathbb{0} &= (), \\ \mathbb{1} &= (1), \end{aligned}$$

where $\oplus$ and $\odot$ are the standard polynomial addition and multiplication operations.

THEOREM 5.1. *The independence polynomial [33, 22] of a graph $G = (V, E)$ can be computed in time $O(|V| \log(|V|)) cc(\mathcal{N}_{\text{IS}}(G))$.*

Proof. By doing the following replacement of tensor elements from the standard number type to the polynomial number type

$$(5.3) \qquad \begin{cases} 1 & \to \mathbb{1}, \\ 0 & \to \mathbb{0}, \\ x_v^{w_v} & \to (0, 1), \end{cases}$$

the tensors W and B, which are introduced in the previous section (Section 4), can thus be written as

$$(5.4) \qquad \left(W^{\text{PN}} \right)^{(v)} = \begin{pmatrix} \mathbb{1} \\ (0, 1) \end{pmatrix}, \qquad \left(B^{\text{PN}} \right)^{(u,v)} = \begin{pmatrix} \mathbb{1} & \mathbb{1} \\ \mathbb{1} & \mathbb{0} \end{pmatrix}.$$

By contracting the tensor network with this polynomial type, we have the exact representation of the independence polynomial. In Equation (5.2: PN), the addition can be computed in time $O(k)$, where $k = \max(k_a, k_b)$, and the multiplication can be evaluated in time $O(k \log(k))$ using the convolution theorem [52]. Since k is upper bounded by the maximum independent set size $\alpha(G) \leq |V|$, the time complexity of element-wise addition and multiplication operation is upper bounded by $|V| \log(|V|)$. Combining with Theorem 4.1, the overall time complexity is $O(|V| \log(|V|)) cc(\mathcal{N}_{\text{IS}}(G))$. $\qquad \square$

265   In practice, using the polynomial type suffers a space overhead proportional to $\alpha(G)$
266 because each polynomial requires a vector of such size to store the coefficients. One may
267 argue that one can first evaluate this polynomial at different x being a real number, and then
268 apply the Gaussian elimination procedure to fit the coefficients of this polynomial. However,
269 in practice this seemingly more time and space-efficient approach suffers from precision
270 issues. The data ranges of standard integer types are too small to cover many practical use
271 cases, while the floating-point numbers may have round-off errors that are much larger than
272 the value itself. These are because the number of independent sets at different sizes may
273 vary by tens or even hundreds of orders of magnitude. For practical methods to evaluate
274 these coefficients, we refer readers to Appendix D, where we provide an accurate and
275 memory-efficient method to find the polynomial by contracting and fitting on finite field
276 algebra. For simplicity, we use this less efficient polynomial algebra for the discussion in the
277 main text.

278   **6. Maximum independent sets and its counting.**

279   **6.1. The number of independent sets.**

280   THEOREM 6.1. *Let $G = (V, E)$ be a graph. The total number independent sets of G can be*
281 *computed in time $cc(\mathcal{N}_{\mathrm{IS}}(G))$.*

282   *Proof.* Let $x = 1$ be a real constant number; the independence polynomial in the previous
283 section becomes

284   (6.1)
$$I(G, 1) = \sum_{k=0}^{\alpha(G)} a_k,$$

285 which corresponds to the total number of independent sets. Since the real number addition
286 and multiplication can be computed in $O(1)$ time, thus proving the theorem by Theorem 4.1. □

287   **6.2. Tropical algebra for finding the MIS size and counting MISs.** Let $x = \infty$, the
288 independence polynomial in the previous section becomes

289   (6.2)
$$I(G, \infty) = \lim_{x \to \infty} \sum_{k=0}^{\alpha(G)} a_k x^k = a_{\alpha(G)} \infty^{\alpha(G)},$$

290 where all terms except the one with the largest order vanish. We can thus replace the polyno-
291 mial type $a = (a_0, a_1, \ldots, a_k)$ with a new type that has two fields: the largest exponent k and
292 its coefficient a_k. From this, we can define a new algebra as

293   (6.3: P1)
$$a_x \infty^x \oplus a_y \infty^y = \begin{cases} (a_x + a_y) \infty^{\max(x,y)}, & x = y \\ a_y \infty^{\max(x,y)}, & x < y \\ a_x \infty^{\max(x,y)}, & x > y \end{cases}$$
$$a_x \infty^x \odot a_y \infty^y = a_x a_y \infty^{x+y}$$
$$\mathbb{0} = 0 \infty^{-\infty}$$
$$\mathbb{1} = 1 \infty^0.$$

294 Here, we have generalized the previous polynomial to the Laurent polynomial to define the
295 zero-element properly. To implement this algebra programmatically, we create a data type
296 with two fields (x, a_x) to store the MIS size and its counting, and define the above operations
297 and constants correspondingly. If one is only interested in finding the MIS size, one can drop

9

the counting field. The algebra of the exponents becomes the max-plus tropical algebra [42, 45]:

$$\begin{aligned}
x \oplus y &= \max(x, y) \\
x \odot y &= x + y \\
\mathbb{0} &= -\infty \\
\mathbb{1} &= 0.
\end{aligned}$$

(6.4: T)

Algebra Equation (6.4: T) and Equation (6.3: P1) are the same as those used in Liu et al. [41] to compute the spin glass ground state energy and its degeneracy.

THEOREM 6.2. *Let $G = (V, E)$ be a graph. Its maximum independent set size $\alpha(G)$ can be computed in time $cc(\mathcal{N}_{\mathrm{IS}}(G))$.*

Proof. By replacing the tensor elements from the standard number type to the tropical numbers in Equation (6.4: T), the vertex and edge tensors transforms to

(6.5)
$$\left(W^{\mathrm{T}}\right)^{(v)} = \begin{pmatrix} \mathbb{1} \\ \infty^{w_v} \end{pmatrix}, \qquad \left(B^{\mathrm{T}}\right)^{(u,v)} = \begin{pmatrix} \mathbb{1} & \mathbb{1} \\ \mathbb{1} & \mathbb{0} \end{pmatrix}.$$

The maximum independent set can be obtained by contracting a tensor network with the above vertex tensors and edge tensors. Since the element-wise addition and multiplication can be computed in $O(1)$ time, the complexity of contracting this tensor network is $cc(\mathcal{N}_{\mathrm{IS}}(G))$ by Theorem 4.1. □

6.3. Truncated polynomial algebra for counting independent sets of large size. Instead of counting just the MISs, one may also be interested in counting the independent sets with the largest several sizes. For example, if one is interested in counting only $a_{\alpha(G)}$ and $a_{\alpha(G)-1}$, we can define a truncated polynomial algebra by keeping only the largest two coefficients in the polynomial in Equation (5.2: PN) as:

(6.6: P2)
$$\begin{aligned}
a \oplus b &= (a_{\max(k_a,k_b)-1} + b_{\max(k_a,k_b)-1}, a_{\max(k_a,k_b)} + b_{\max(k_a,k_b)}), \\
a \odot b &= (a_{k_a-1}b_{k_b} + a_{k_a}b_{k_b-1}, a_{k_a}b_{k_b}), \\
\mathbb{0} &= (), \\
\mathbb{1} &= (1).
\end{aligned}$$

In the program, we thus need a data structure that contains three fields, the largest order k, and the coefficients for the two largest orders a_k and a_{k-1}. This approach can clearly be extended to calculate more independence polynomial coefficients and is more efficient than calculating the entire independence polynomial. Similarly, one can also truncate the polynomial and keep only its smallest several orders. It can be used, for example, to count the maximal independent sets with the smallest cardinality, where a maximal independent set is an independent set that cannot be made larger by adding a new vertex into it without violating the independence constraint. As will be shown below, this algebra can also be extended to enumerate those large-size independent sets.

THEOREM 6.3. *The number of the largest K independent sets of a graph $G = (V, E)$ can be computed in time $O(K \log K)cc(\mathcal{N}_{\mathrm{IS}}(G))$.*

The proof is similar to the proof of Theorem 5.1, except only K largest coefficients are involved in the addition and multiplication.

7. Enumerating and sampling independent sets.

10

7.1. Set algebra for configuration enumeration. The configuration enumeration of independent sets include, for example, the enumeration of all independent sets, the enumeration of all MISs, and the enumeration of independent sets with the largest several sizes. Recall that in the definition of a vertex tensor in Equation (4.2), variables carry labels, so that one can read out all independent sets directly from the output polynomials. The multiplication between labelled variables is commutative while the summation of labelled variables forms a set. Intuitively, one can use a bit string as the representation of a labelled variable and use the bit-wise or $\vee^\circ$ as the multiplication operation. For example, in a 5-vertex graph, x_2 and x_5 can be represented as 01000 and 00001 respectively and their multiplication $x_2 x_5$ can be represented as $01000 \vee^\circ 00001 = 01001$. To enumerate all independent sets, we designed an algebra on sets of bit strings:

(7.1: SN)
$$
\begin{aligned}
s \oplus t &= s \cup t \\
s \odot t &= \{\sigma \vee^\circ \tau \mid \sigma \in s, \tau \in t\} \\
\mathbb{0} &= \{\} \\
\mathbb{1} &= \{0^{\otimes |V|}\},
\end{aligned}
$$

where s and t are each a set of $|V|$-bit strings.

Example 4. For elements that are bit strings of length 5, we have the following set algebra

$$\{00001\} \oplus \{01110, 01000\} = \{01110, 01000\} \oplus \{00001\} = \{00001, 01110, 01000\}$$
$$\{00001\} \oplus \{\} = \{00001\}$$

$$\{00001\} \odot \{01110, 01000\} = \{01110, 01000\} \odot \{00001\} = \{01111, 01001\}$$
$$\{00001\} \odot \{\} = \{\}$$
$$\{00001\} \odot \{00000\} = \{00001\}.$$

LEMMA 7.1. *The independent sets of a graph $G = (V, E)$ can be enumerated in time $O(|I| * |V|) cc(\mathcal{N}_{\mathrm{IS}}(G))$, where $|I|$ is the number of independent sets.*

Proof. To enumerate over I, we initialize the variable x_i in the vertex tensor to $x_i = \{\boldsymbol{e}_i\}$, where $\boldsymbol{e}_i$ is a basis bit string of size $|V|$ that has only one non-zero value at location i. The vertex and edge tensors are thus

(7.2)
$$
\left(W^{\mathrm{SN}}\right)^{(v)} = \begin{pmatrix} \mathbb{1} \\ \{\boldsymbol{e}_i\} \end{pmatrix}, \qquad \left(B^{\mathrm{SN}}\right)^{(u,v)} = \begin{pmatrix} \mathbb{1} & \mathbb{1} \\ \mathbb{1} & \mathbb{0} \end{pmatrix}.
$$

The contraction of this tensor network is a set with its elements being all independent set configurations. The time complexity of element-wise addition and multiplication is upper bounded by $|I| * |V|$, where $|V|$ comes from the linear cost of representing a $|V|$-vertex configuration. Combining with Theorem 4.1, the overall time complexity is $O(|I| * |V|) cc(\mathcal{N}_{\mathrm{IS}}(G))$. In practice, the huge multiplicative factor $|I|$ only appear in the last step of tensor contraction. This complexity only serves as an upper bound for the actual performance of the algorithm. □

This set algebra can serve as the coefficients in Equation (5.2: PN) to enumerate independent sets of all different sizes, Equation (6.3: P1) to enumerate all MISs, or Equation (6.6: P2) to enumerate all independent sets of sizes $\alpha(G)$ and $\alpha(G) - 1$. As long as

364 the coefficients in a truncated polynomial are members of a commutative semiring, the
365 polynomial itself is a commutative semiring.
366   For example, to enumerate only the MISs, we can define a combined element type $s_k \infty^k$,
367 where the coefficients follow the algebra in Equation (7.1: SN) and the exponents follow the
368 max-plus tropical algebra. The combined operations become:

369 (7.3: P1+SN)

$$s_x \infty^x \oplus s_y \infty^y = \begin{cases} (s_x \cup s_y) \infty^{\max(x,y)}, & x = y \\ s_y \infty^{\max(x,y)}, & x < y \\ s_x \infty^{\max(x,y)}, & x > y \end{cases},$$

$$s_x \infty^x \odot s_y \infty^y = \{\sigma \vee^\circ \tau \mid \sigma \in s_x, \tau \in s_y\} \infty^{x+y},$$

$$\mathbb{0} = \{\} \infty^{-\infty},$$

$$\mathbb{1} = \{0^{\otimes |V|}\} \infty^0.$$

370   LEMMA 7.2. *Let $G = (V, E)$ be a graph and $\mathcal{I}_\alpha$ be the set of all maximum independent*
371 *sets of G. $\mathcal{I}_\alpha$ can be computed in time $O(|\mathcal{I}_\alpha| * |V|)cc(\mathcal{N}_{\text{IS}}(G))$.*

372   *Proof.* By replacing the tensor elements in Section 4 with the number type defined in
373 Equation (7.3: P1+SN) and with variable $x_v^{w_v} = \{e_v\} \infty^{w_v}$, the vertex tensor and edge tensor
374 become

375 (7.4) $$\left(W^{\text{P1+SN}}\right)^{(v)} = \begin{pmatrix} \mathbb{1} \\ \{e_v\} \infty^{w_v} \end{pmatrix}, \qquad \left(B^{\text{P1+SN}}\right)^{(u,v)} = \begin{pmatrix} \mathbb{1} & \mathbb{1} \\ \mathbb{1} & 0 \end{pmatrix}.$$

376 The enumeration of all MIS configurations corresponds to the contraction of this tensor
377 network. The time complexity of element-wise addition and multiplication is upper bounded
378 by the number of maximum independent sets $|\mathcal{I}_\alpha|$. Combining with Theorem 4.1, the overall
379 time complexity is $O(|\mathcal{I}_\alpha| * |V|)cc(\mathcal{N}_{\text{IS}}(G))$. $\qquad\qquad\square$

380 However, direct contraction might have significant space overheads for keeping too many
381 intermediate states irrelevant to the final maximum independent sets. We introduce the
382 bounding technique in Appendix C to avoid this issue. One may also be interested in the
383 more studied maximal independent sets [12, 20, 35] enumeration. We discuss this in
384 Appendix B.1 since it requires using a different tensor network structure.

385   LEMMA 7.3. *Let $G = (V, E)$ be a graph and $\mathcal{I}_k$ be the set of independent sets with size k.*
386 *The independent sets with size $k > \alpha(G) - K$ of G can be computed in time*
387 $O\left(|V| * \sum_{k=\alpha(G)-K+1}^{\alpha(G)} |\mathcal{I}_k|\right)cc(\mathcal{N}_{\text{IS}}(G))$.

388   *Proof.* This can be proved by combining the set algebra Equation (7.1: SN) with the
389 polynomial number truncated to largest K orders (Equation (6.6: P2)). The number of
390 bitstring operations in the addition and multiplication of the joint algebra is upper bounded
391 by the elements in the sets $\sum_{k=\alpha(G)-K+1}^{\alpha(G)} |\mathcal{I}_k|$. Combining with Theorem 4.1, the overall time
392 complexity of contracting this tensor network is $O\left(|V| * \sum_{k=\alpha(G)-K+1}^{\alpha(G)} |\mathcal{I}_k|\right)cc(\mathcal{N}_{\text{IS}}(G))$. $\quad\square$

393   If one is interested in obtaining only one MIS configuration, they can just keep one
394 configuration in each tensor element to save the computational effort. By replacing the sets
395 of bit strings in Equation (7.1: SN) with a single bit string, we have the following algebra

396 (7.5: S1)

$$\sigma \oplus \tau = \texttt{select}(\sigma, \tau),$$

$$\sigma \odot \tau = (\sigma \vee^\circ \tau),$$

$$\mathbb{0} = 1^{\otimes |V|},$$

$$\mathbb{1} = 0^{\otimes |V|}.$$

12

The `select` function picks one of σ and τ by some criteria. It can be picking the one smaller in the lexicographical order such that the addition operation is commutative and associative. In most cases, it is completely fine for the `select` function to pick a random one (not commutative and associative anymore) to generate a random MIS.

Proposition 7.4. *Let $G = (V, E)$ be a graph. One of its maximum independent set can be computed in time $O(|V|)cc(\mathcal{N}_{IS}(G))$.*

The proof is similar to Lemma 7.2, except the set algebra is replace by the one in Equation (7.5: S1), and the time complexity of element-wise addition and multiplication no longer depends on $|\mathcal{I}_\alpha|$. The linear dependency of $|V|$ in the time complexity can be remove using the back propagation technique, at the cost of additional space.

Theorem 7.5. *Let $G = (V, E)$ be a graph. One of its maximum independent set can be computed in time $cc(\mathcal{N}_{IS}(G))$.*

Proof. Let $\alpha(G) = \max\limits_{s \in I} \sum_v s_v$ be the maximum independent set size. The optimal configuration can be obtained by differentiating over vertex tensors as

(7.6)
$$\begin{cases} 1, & \frac{\partial \alpha(G)}{\partial s_v} = 1 \\ 0, & \frac{\partial \alpha(G)}{\partial s_v} = 0 \end{cases}.$$

It can be numerically computed by differentiating over the process of the tropical tensor network contraction, where the backward rules can be found in Appendix C. The time overhead of computing a single optimal solution is constant compared to only computing the contraction. Thus, the computing time complexity should be the same as in Theorem 6.2. □

7.2. Sampling from the extremely large configuration space. When the problem size becomes larger, a set of all bitstrings might be impossible to fit into any type of storage. To get something meaningful out of the configuration space, we use a binary sum-product expression tree as a compact representation of a set of configurations, i.e. instead of directly computing a set using the algebra in Equation (7.1: SN), we store the process of computing it. Each node in this tree is a quadruple $(type, data, left, right)$, where $type$ is one of `LEAF`, `ZERO`, `SUM` and `PROD`, $data$ is a bit string as the content in a `LEAF` node, and $left$ and $right$ are left and right operands of a `SUM` or `PROD` node.

(7.7: EXPR)
$$\begin{aligned} s \oplus t &= (\texttt{SUM}, /, s, t) \\ s \odot t &= (\texttt{PROD}, /, s, t) \\ \mathbb{0} &= (\texttt{ZERO}, /, /, /) \\ \mathbb{1} &= (\texttt{LEAF}, 0^{\otimes |V|}, /, /). \end{aligned}$$

This algebra is a commutative semiring because we define the equivalence of two sum-product expression trees by comparing their expanded (using Equation (7.1: SN)) forms rather than their storage. Except using the sum-product expression tree directly as tensor elements, one can also let it be the coefficients of a (truncated) polynomial to compute such trees for independent sets with the largest several sizes.

Theorem 7.6. *The independent sets I of a graph $G = (V, E)$ can be enumerated in time $cc(\mathcal{N}_{IS}(G)) + O(|V| * |I|)$.*

Proof. By replacing the tensor elements with the number type in Equation (7.7: EXPR) with $x_v^{w_v}$ mapped to $(\texttt{LEAF}, \boldsymbol{e}_i, /, /)$, the vertex tensor and edge tensor become

(7.8)
$$\left(W^{\text{EXPR}}\right)^{(v)} = \begin{pmatrix} \mathbb{1} \\ (\texttt{LEAF}, \boldsymbol{e}_i, /, /) \end{pmatrix}, \qquad \left(B^{\text{EXPR}}\right)^{(u,v)} = \begin{pmatrix} \mathbb{1} & \mathbb{1} \\ \mathbb{1} & \mathbb{0} \end{pmatrix}.$$

13

The contraction result of this tensor network corresponds to a sum-product expression tree for the set of independent sets. Unlike in Lemma 7.1, the contraction complexity is independent of the set size, i.e. the time complexity is $cc(\mathcal{N}_{\mathrm{IS}}(G))$ by Theorem 4.1. The time complexity to evaluating this expression tree is $O(|V| * |I|)$, where $|V|$ comes from the number of bits to represent a configuration. Proving the theorem by adding two computing times. □

Similarly, we have the following corollary by combining the sum-product expression tree algebra with the truncated polynomial.

COROLLARY 7.7. *Let $G = (V, E)$ be a graph and $\mathcal{I}_k$ be the number of independent sets with size k. The independent sets with size $k > \alpha(G) - K$ can be computed in time $O(K \log K)cc(\mathcal{N}_{\mathrm{IS}}(G)) + O\left(|V| * \sum_{k=\alpha(G)-K+1}^{\alpha(G)} |\mathcal{I}_k|\right)$.*

Again, the theorem can be proven by relating it with its set algebra version in Lemma 7.3. Because it is unlikely that one can collect all configurations represented by a sum-product expression tree into a set due to its space complexity, one can also use this construction to produce unbiased samples of the sum-product tree.

THEOREM 7.8. *Let $G = (V, E)$ be a graph. Its independent sets with size $k > \alpha(G) - K$ can be unbiasedly sampled in time $cc(\mathcal{N}_{\mathrm{IS}}(G)) + O(|V| * |E| * M)$, where M is the number of samples.*

Proof. Similar to the proof of Theorem 7.6, we first obtain a sum-product tree representation of the configurations in time $cc(\mathcal{N}_{\mathrm{IS}}(G))$. Then we generate a sample using the following procedure with cost $O(|V| * |E|)$. The sampling program starts from the root node and descends this tree recursively to the left and right siblings. If a node has type SUM, the program draws samples from the left and right siblings with a probability decided by the size of each sub-tree and returns the union of samples. Otherwise, if a node has type PROD, the program draws two sets of samples of equal sizes from its left and right siblings and returns the element-wise multiplication ($\vee°$) of them. The recursion stops at a LEAF node having size 1 or a ZERO node having size 0. In a sum-product expression tree, the number of configurations of sub-trees can be determined easily as we will show in the following example.

Example 5. Let us consider the following sum-product expression tree

$$(\mathtt{SUM}, /, (\mathtt{PROD}, /, A, B), (\mathtt{SUM}, /, C, D)),$$

where sub-trees A, B, C and D can be any of the four types of nodes. This sum-product expression tree can be represented diagrammatically as the following:
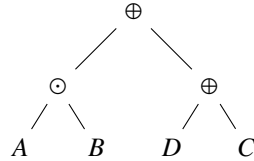
The left and right siblings of the root node have sizes $|A| * |B|$ and $|C| + |D|$ respectively, while the root node size can be computed as $|A| * |B| + |C| + |D|$. The sizes of $A, B, C,$ and D can be computed recursively until the program meets either a LEAF node or a ZERO node, which has a known size of 1 or 0.

Since the depth of the expression tree is $O(|E|)$ and the cost of each arithmetic operation is $O(|V|)$, the overall time complexity to generate a sample is $O(|V| * |E|)$, hence proving the theorem. □

14

Similarly, if one is only interested in obtaining independent sets with largest K sizes, we have the following corollary.

COROLLARY 7.9. *Let $G = (V, E)$ be a graph. Its independent sets with size $k > \alpha(G) - K$ can be unbiasedly sampled in time $O(K \log K)cc(\mathcal{N}_{\mathrm{IS}}(G)) + O(|V| * |E| * M)$, where M is the number of samples.*

8. Weighted graphs. All the solution space properties and the corresponding algebra on unweighted graphs still hold for integer-weighted graphs, while for general weighted graphs, the independence polynomial is not well defined anymore. For general weighted graphs, it is more useful to know the k maximum weighted sets and their sizes. They can be computed by the extended tropical algebra, which is a natural generalization of the max-plus tropical algebra:

(8.1: Tk)
$$s \oplus t = \texttt{largest}(s \cup t, k)$$
$$s \odot t = \texttt{largest}(\{a + b \mid a \in s, b \in t\}, k)$$
$$\mathbb{0} = -\infty^{\otimes k}$$
$$\mathbb{1} = -\infty^{\otimes k-1} \otimes 0$$

where $\texttt{largest}(s, k)$ means truncating the set s by only keeping its k largest values.

THEOREM 8.1. *Let $G = (V, E)$ be a weighted graph. Its K largest independent set sizes can be computed in time $O(K \log K)cc(\mathcal{N}_{\mathrm{IS}}(G))$.*

Proof. The K largest independent sets can be computed by contracting a tensor network with extended tropical numbers. The vertex tensor and edge tensor are

(8.2)
$$\left(W^{\mathrm{T}k}\right)^{(v)} = \begin{pmatrix} \mathbb{1} \\ -\infty^{\otimes k-1} \otimes w_v \end{pmatrix}, \qquad \left(B^{\mathrm{T}k}\right)^{(u,v)} = \begin{pmatrix} \mathbb{1} & \mathbb{1} \\ \mathbb{1} & \mathbb{0} \end{pmatrix},$$

where we have used $x_v^{w_v} = (-\infty^{\otimes k-1} \otimes 1)^{w_v} = -\infty^{\otimes k-1} \otimes w_v$.

The computation of $s \odot t$ is a maximum sum combination problem that can be done in time $O(k \log(k))$ using the algorithm in Section SM4. Hence, the overall time complexity of contracting the tensor network is $O(K \log K)cc(\mathcal{N}_{\mathrm{IS}}(G))$. □

COROLLARY 8.2. *Let $G = (V, E)$ be a weighted graph. Its K largest independent sets can be computed in time $O(|V| * K \log K)cc(\mathcal{N}_{\mathrm{IS}}(G))$.*

To find solutions corresponding to the largest K sizes, one can combine the extended tropical algebra with the bit-string algebra (Equation (7.5: S1)). Since the $\oplus$ operation of the configuration sampler is not used in the combined algebra, the resulting configurations are deterministic and complete.

9. Example applications.

9.1. Number of independent sets and entropy constant for hardcore lattice gases. We compute the counting of all independent sets for graphs shown in Figure 2, where vertices are all placed on square lattices of dimensions $L \times L$. The types of graphs include: the square lattice graphs (Figure 2(a)), the square lattice graphs with a filling factor $p = 0.8$, which means $\lfloor pL^2 \rceil$ sites are occupied with vertices (Figure 2(b)), the King's graphs (Figure 2(c)), the King's graphs with a filling factor $p = 0.8$ (Figure 2(d)), which is the ensemble of graphs used in Ref. [19] to benchmark quantum algorithms on a Rydberg atom array quantum computer.

The number of independent sets for square lattice graphs of size $L \times L$ form a well-known integer sequence (OEIS A006506), which is thought as a two-dimensional generalization of
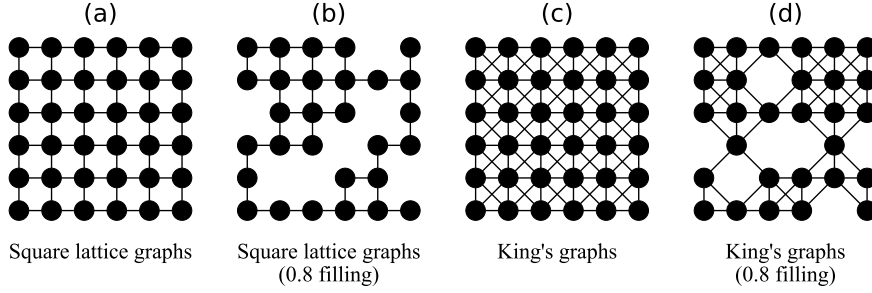
15

Figure 2: The types of graphs used in the case study in Subsection 9.1. The lattice dimensions are $L \times L$. (a) square lattice graphs. (b) square lattice graphs with a filling factor $p = 0.8$. (c) King's graphs. (d) King's graphs with a filling factor $p = 0.8$.

the Fibonacci numbers. We computed the integer sequence for $L = 38$ and $L = 39$, which is, to the best of our knowledge, not known before. In the computation, we used finite-field algebra for contracting integer tensor networks with arbitrarily high precision.

A theoretically interesting number that can be computed using the number of independent sets is the entropy constant, which can describe the thermodynamic properties of hard-core lattice gases at the high-temperature limit. For the square lattice graphs, this number is called the *hard square entropy constant* (OEIS A085850), which is defined as $\lim_{L \to \infty} F(L, L)^{1/L^2}$, where $F(L, L)$ is the number of independent sets of a given lattice dimensions $L \times L$. This quantity arises in statistical mechanics of hard-square lattice gases [5, 49] and is used to understand phase transitions for these systems. This entropy constant is not known to have an exact representation, but it is accurately known in many digits. Similarly, we can define entropy constants for other lattice gases. In Figure 3, we look at how $F(L, L)^{1/\lfloor pL^2 \rceil}$ scales as a function of the grid size L for all types of graphs shown in Figure 2. Our results match the known results for the non-disordered square lattice and King's graphs. For disordered square lattice and King's graphs with a filling factor $p = 0.8$, we randomly sample 1000 graph instances. To our knowledge, the entropy constants for these disordered graphs have not been studied before. They may be used to study phase transitions for disordered lattices, which are typically much harder to understand. Interestingly, the variations due to different random instances are negligible for this quantity.

9.2. The overlap gap property. With this tool to enumerate or sample configurations, one can understand the structure of the independent set configuration space, such as the optimization landscape for finding the MISs. One of the known barriers to finding the MIS is the so-called overlap gap property [26, 25]. If the overlap gap property is present, it means every two large independent sets either have a significant intersection or a very small intersection; it implies that large independent sets are clustered together. This clustering property has been used to rigorously prove upper bounds on the performance of local search algorithms [26, 25]. To investigate the overlap gap property, we compute pair-wise Hamming distance distributions of large independent sets as they are good indicators of the presence or absence of overlap gap properties. We inspect two types of graphs that are particularly interesting, the King's graphs with defects and 3-regular graphs. It is known that the MIS problem on a general graph can be mapped to the King's graph with defects [27, 19]. However, it is not clear whether the MIS problem defined on a randomly generated King's graph with defects can have the overlap gap property. It is known that
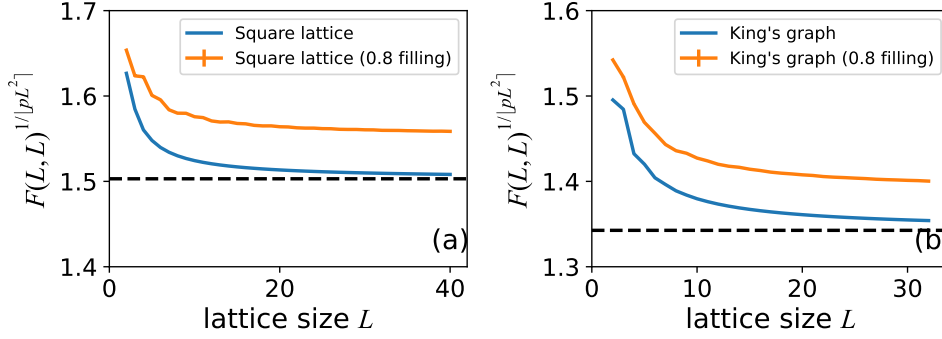
16

Figure 3: Mean entropy for lattice gases on graphs defined in Figure 2. We sampled 1000 instances for $p = 0.8$ lattices and the error bar is too small to be visible. The horizontal black dashed lines are for $\lim_{L\to\infty} F(L,L)^{1/L^2}$ for the corresponding non-disordered square lattice and King's graphs.

finding MISs of a d-regular graphs has the overlap gap property [51, 24] when both d and the graph sizes are large, but, it is not known whether, for small d, e.g. for 3-regular graphs, this statement remains true. We randomly generated 9 instances for each category of King's graph at 0.8 filling with dimensions 20×20 (320 vertices) and 3-regular graphs with 110 vertices. At this problem size, there are too many independent sets to fit into any storage, hence we combine the truncated polynomial and sum-product expression tree to directly sample from the target configuration space. For each instance G, we sample 10^4 pairs of configurations from the independent sets of sizes $\geq \lceil \gamma \times \alpha(G) \rceil$ and show the pair-wise Hamming distance distribution in Figure 4. We observe a clear single peak structure at a fixed distance normalized by the MIS size for the King's graphs, indicating the absence of the overlap gap property in a random King's graph at 0.8 filling. Since the MIS problem on an arbitrary graph can be mapped to a King's graph at a certain filling, this result is highly nontrivial. It likely implies that the King's graphs with defects mapped from hard MIS instances have a very small measure in the total defected King's graph space. In contrast, very different pair-wise Hamming distributions are obtained in Figure 4(b), where we observed the multiple peak structure when the control parameter γ is big enough. It indicates the existence of disconnected clusters in the configuration space of the MIS problem on 3-regular graphs. We expect this numerical tool can be used to understand this phenomenon better and to further investigate the graph properties and the geometry of the configuration spaces for a variety of graph instances.

9.3. Analyzing quantum and classical algorithms for Maximum Independent Set.
In a recent work, the ability to enumerate configurations and compute independence polynomials was critical in understanding the performance of quantum optimization algorithms for the MIS problem on a Rydberg atom quantum computer [19]. This work focused on exploring King's graphs with 0.8 filling. The hardest instances for classical simulated annealing could be accurately predicted from the independence polynomial, which gave information about the density of local minima at different independent set sizes. On the hardest graph instances for simulated annealing studied in the experiment, a high density of local minima were found at independent set sizes of $\alpha(G) - 1$, which the algorithm became trapped in instead of finding the optimal solution of size $\alpha(G)$. By

17

$\gamma = 0.9$

$\gamma = 0.95$

$\gamma = 0.975$

(a) Nine King's graphs of size 20 ×20, 0.8 filling.

$\gamma = 0.9$

$\gamma = 0.95$

$\gamma = 0.975$

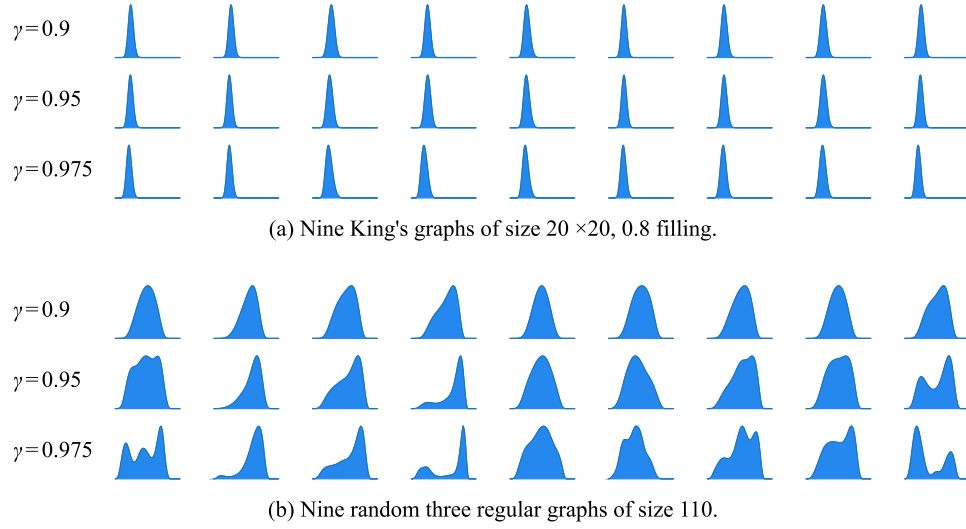(b) Nine random three regular graphs of size 110.

Figure 4: Pairwise Hamming distances distribution for configurations sampled from independent sets with sizes $\geq \lceil \gamma \times \alpha(G) \rceil$. In each plot, the x-axis is the Hamming distance normalized by the total number of vertices and the y-axis is the probability.
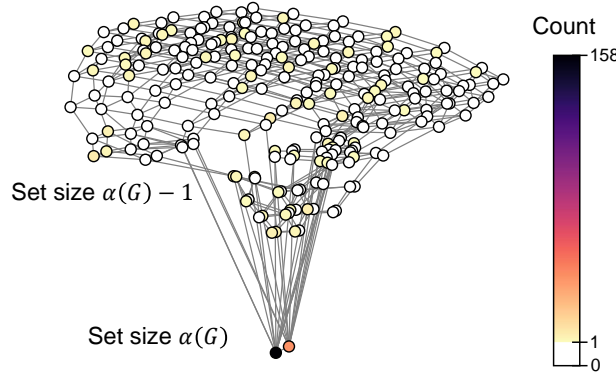

Figure 5: Visualization of experimental outputs of a quantum algorithm for solving the maximum independent set problem [19]. Each vertex represents an independent set, and each edge represents a pair of independent sets that differ by a swap operation or a vertex addition/removal.

enumerating the configurations using techniques described in the present work, we found that simulated annealing randomly explores the independent sets of size $\alpha(G) - 1$ until an optimum solution is found. Therefore, the large ratio of local to global minima prevents simulated annealing from efficiently finding an MIS.

Although the performance of the quantum algorithm is more challenging to understand due to the inherent difficulty in studying quantum systems, the present methods allow one to gain significant insights by visualizing the experimental outputs of a quantum algorithm over

18

the solution space, as shown in Figure 5 for instances with 39 nodes [19]. Here, the structure of the solution space is shown on a graph where each vertex represents a large independent set. Each edge represents a pair of independent sets that differ by a swap operation or a vertex addition to the set, which are the operations naturally present in the effective dynamics at the end of the quantum algorithm. The solution space graph is well-connected by local changes to the spin configurations, has a small diameter, and the degree of each node appears to concentrate. This visualization makes it clear that the quantum algorithm does not appear to return solely local minima with a large Hamming distance from the MISs as suggested for adiabatic algorithms e.g. by Ref. [4] which would appear as a long path on the solution space graph from the sampled local minima to the MIS. Instead, the quantum algorithm samples from local minima across the solution space graph with a wide range of Hamming distances from the MISs. In the case where a large superposition state of local minima is created during the coherent evolution, the quantum algorithm achieves a quadratic speedup over simulated annealing [19]. Looking forward, we expect these tools can be applied to understanding the performance of quantum and classical algorithms on a wide class of NP-hard combinatorial optimization problems.

10. Discussion and conclusion. In this work, we introduced a framework that uses generic tensor networks to compute different solution space properties of a certain class of NP-hard combinatorial optimization problems. Each solution space property is computed using the same tensor network with different tensor element algebra. The different data types introduced in the main text to compute these properties are summarized in the diagram in Figure 1. The class of problems solvable by a tensor network includes but is not limited to maximum independent sets and a variety of other combinatorial problems such as the matching problem, the k-coloring problem, the max-cut problem, the set packing problem, and the set covering problem, as detailed in Appendix B.

Looking ahead, it could be possible to generalize the idea of generic programming to other algorithms that have certain algebraic structures such as those using the inclusion-exclusion principle or subset convolution [23] and explore what new properties can be computed. To this end, dynamic programming [17, 23] approaches could be considered. Dynamic programming is closely related to a tropical tensor network [41]; for example, the Viterbi algorithm for finding the most probable configuration in a hidden Markov model can be interpreted as a matrix product state featured with tropical algebra, and the tropical tensor network in the main text is potentially equivalent to dynamic programming in finding an optimum solution. Since dynamic programming has much broader applications, it would be interesting to extend the ideas from this paper to provide an algebraic interpretation for dynamic programming so that it can be used to compute other solution space properties beyond just finding an optimum solution. It is also possible to extend this idea to other algebras. For example, generic semiring algebra has been used in computational linguistics to compute interesting quantities of a given grammar and string [30].

The source code in the Julia language for this paper can be found in the Github repository [1]. There is a short introduction to this repository as well a gist to show how it works in Section SM5 as well. We expect our tool can be used to understand and study many interesting applications of independent sets and beyond. We also hope the toolkit we built, including tensor network contraction order optimization and efficient tropical matrix multiplication, can be helpful to the development of other scientific software.

19

REFERENCES

[1] https://github.com/QuEraComputing/GenericTensorNetworks.jl.

[2] https://github.com/TensorBFS/TropicalGEMM.jl.

[3] S. Alikhani and Y. hock Peng, *Introduction to domination polynomial of a graph*, 2009, https://arxiv.org/abs/0905.2251.

[4] B. Altshuler, H. Krovi, and J. Roland, *Anderson localization makes adiabatic quantum optimization fail*, Proceedings of the National Academy of Sciences, 107 (2010), pp. 12446–12450, https://doi.org/10.1073/pnas.1002116107.

[5] R. J. Baxter, I. G. Enting, and S. K. Tsang, *Hard-square lattice gas*, Journal of Statistical Physics, 22 (1980), pp. 465–489, https://doi.org/10.1007/BF01012867.

[6] T. Besard, C. Foket, and B. De Sutter, *Effective extensible programming: unleashing julia on gpus*, IEEE Transactions on Parallel and Distributed Systems, 30 (2018), pp. 827–841, 10.1109/TPDS.2018.2872064.

[7] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, *Julia: A fast dynamic language for technical computing*, 2012, https://arxiv.org/abs/1209.5145.

[8] J. Biamonte and V. Bergholm, *Tensor networks in a nutshell*, 2017, https://arxiv.org/abs/1708.00006.

[9] J. D. Biamonte, J. Morton, and J. Turner, *Tensor network contractions for #SAT*, Journal of Statistical Physics, 160 (2015), pp. 1389–1404, https://doi.org/10.1007/s10955-015-1276-z, https://doi.org/10.1007%2Fs10955-015-1276-z.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006, https://link.springer.com/gp/book/9780387310732.

[11] M. Bousquet-Mélou, S. Linusson, and E. Nevo, *On the independence complex of square grids*, Journal of Algebraic combinatorics, 27 (2008), pp. 423–450, https://doi.org/10.1007/s10801-007-0096-x.

[12] C. Bron and J. Kerbosch, *Algorithm 457: finding all cliques of an undirected graph*, Communications of the ACM, 16 (1973), pp. 575–577, https://doi.org/10.1145/362342.362367.

[13] S. Butenko and P. M. Pardalos, *Maximum Independent Set and Related Problems, with Applications*, PhD thesis, USA, 2003, https://ufdc.ufl.edu/UFE0001011/00001.

[14] P. Butera and M. Pernici, *Sums of permanental minors using grassmann algebra*, 2014, https://arxiv.org/abs/1406.5337.

[15] A. Cichocki, *Era of big data processing: A new approach via tensor networks and tensor decompositions*, 2014, https://doi.org/10.48550/ARXIV.1403.2048.

[16] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete, *Matrix product states and projected entangled pair states: Concepts, symmetries, theorems*, Reviews of Modern Physics, 93 (2021), https://doi.org/10.1103/revmodphys.93.045003.

[17] B. Courcelle, *The monadic second-order logic of graphs. i. recognizable sets of finite graphs*, Information and computation, 85 (1990), pp. 12–75, https://doi.org/10.1016/0890-5401(90)90043-H.

[18] J. C. Dyre, *Simple liquids' quasiuniversality and the hard-sphere paradigm*, Journal of Physics: Condensed Matter, 28 (2016), p. 323001, https://doi.org/10.1088/0953-8984/28/32/323001.

[19] 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, S.-T. Wang, M. Greiner, V. Vuletić, and M. D. Lukin, *Quantum optimization of maximum independent set using rydberg atom arrays*, Science, 376 (2022), pp. 1209–1215, https://doi.org/10.1126/

science.abo6587, https://www.science.org/doi/abs/10.1126/science.abo6587, https://arxiv.org/abs/https://www.science.org/doi/pdf/10.1126/science.abo6587.

[20] D. Eppstein, M. Löffler, and D. Strash, *Listing all maximal cliques in sparse graphs in near-optimal time*, in Algorithms and Computation, O. Cheong, K.-Y. Chwa, and K. Park, eds., Berlin, Heidelberg, 2010, Springer Berlin Heidelberg, pp. 403–414, https://doi.org/10.1007/978-3-642-17517-6_36.

[21] H. C. M. Fernandes, J. J. Arenzon, and Y. Levin, *Monte carlo simulations of two-dimensional hard core lattice gases*, The Journal of Chemical Physics, 126 (2007), p. 114508, https://doi.org/10.1063/1.2539141.

[22] G. M. Ferrin, *Independence polynomials*, (2014), https://scholarcommons.sc.edu/etd/2609/.

[23] F. V. Fomin and P. Kaski, *Exact exponential algorithms*, Communications of the ACM, 56 (2013), pp. 80–88, https://doi.org/10.1145/2428556.2428575.

[24] D. Gamarnik, *The overlap gap property: A topological barrier to optimizing over random structures*, Proceedings of the National Academy of Sciences, 118 (2021), https://doi.org/10.1073/pnas.2108492118.

[25] D. Gamarnik and A. Jagannath, *The overlap gap property and approximate message passing algorithms for p-spin models*, 2019, https://arxiv.org/abs/1911.06943.

[26] D. Gamarnik and M. Sudan, *Limits of local algorithms over sparse random graphs*, 2013, https://arxiv.org/abs/1304.1831.

[27] M. R. Garey and D. S. Johnson, *The Rectilinear Steiner Tree Problem is NP-Complete*, SIAM Journal on Applied Mathematics, 32 (1977), pp. 826–834, https://doi.org/10.1137/0132071.

[28] S. Gaspers, D. Kratsch, and M. Liedloff, *On independent sets and bicliques in graphs*, Algorithmica, 62 (2012), pp. 637–658, https://doi.org/10.1007/s00453-010-9474-1.

[29] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3, JHU press, 2013, https://doi.org/10.2307/3621013.

[30] J. Goodman, *Semiring parsing*, Computational Linguistics, 25 (1999), pp. 573–606, https://aclanthology.org/J99-4004.pdf.

[31] J. Gray and S. Kourtis, *Hyper-optimized tensor network contraction*, Quantum, 5 (2021), p. 410, https://doi.org/10.22331/q-2021-03-15-410.

[32] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, *Array programming with NumPy*, Nature, 585 (2020), p. 357–362, https://doi.org/10.1038/s41586-020-2649-2.

[33] N. J. Harvey, P. Srivastava, and J. Vondrák, *Computing the independence polynomial: from the tree threshold down to the roots*, in Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2018, pp. 1557–1576, https://doi.org/10.1137/1.9781611975031.102.

[34] J. Hastad, *Clique is hard to approximate within $n^{1-\epsilon}$*, in Proceedings of 37th Conference on Foundations of Computer Science, IEEE, 1996, pp. 627–636, https://doi.org/10.1007/BF02392825.

[35] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou, *On generating all maximal independent sets*, Information Processing Letters, 27 (1988), pp. 119–123, https://doi.org/10.1016/0020-0190(88)90065-8.

[36] G. Kalachev, P. Panteleev, and M.-H. Yung, *Multi-tensor contraction for xeb verification of quantum circuits*, 2021, https://arxiv.org/abs/2108.05665.

[37] L. R. Kerr, *The effect of algebraic structure on the computational complexity of matrix multiplication*, tech. report, Cornell University, 1970, https://ecommons.cornell.edu/handle/1813/5934.

[38] S. Kourtis, C. Chamon, E. Mucciolo, and A. Ruckenstein, *Fast counting with tensor networks*, SciPost Physics, 7 (2019), https://doi.org/10.21468/scipostphys.7.5.060.

[39] T.-D. Lee and C.-N. Yang, *Statistical theory of equations of state and phase transitions. ii. lattice gas and ising model*, Physical Review, 87 (1952), p. 410, https://doi.org/10.1103/PhysRev.87.410.

[40] V. E. Levit and E. Mandrescu, *The independence polynomial of a graph at -1*, 2009, https://arxiv.org/abs/0904.4819.

[41] J.-G. Liu, L. Wang, and P. Zhang, *Tropical tensor network for ground states of spin glasses*, Physical Review Letters, 126 (2021), https://doi.org/10.1103/physrevlett.126.090506.

[42] D. Maclagan and B. Sturmfels, *Introduction to tropical geometry*, vol. 161, American Mathematical Soc., 2015, http://www.cs.technion.ac.il/~janos/COURSES/238900-13/Tropical/MaclaganSturmfels.pdf.

[43] F. Manne and S. Sharmin, *Efficient counting of maximal independent sets in sparse graphs*, in International Symposium on Experimental Algorithms, Springer, 2013, pp. 103–114, https://doi.org/10.1007/978-3-642-38527-8_11.

[44] I. L. Markov and Y. Shi, *Simulating quantum computation by contracting tensor networks*, SIAM Journal on Computing, 38 (2008), p. 963–981, https://doi.org/10.1137/050644756.

[45] C. Moore and S. Mertens, *The nature of computation*, OUP Oxford, 2011, https://doi.org/10.1093/acprof:oso/9780199233212.001.0001.

[46] R. Orús, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Annals of Physics, 349 (2014), pp. 117–158, https://doi.org/10.1016/j.aop.2014.06.013.

[47] I. V. Oseledets, *Tensor-train decomposition*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2295–

750   2317, https://doi.org/10.1137/090752286.

[48] F. Pan and P. Zhang, *Simulating the sycamore quantum supremacy circuits*, 2021, https://arxiv.org/abs/2103.03074.

[49] P. A. Pearce and K. A. Seaton, *A classical theory of hard squares*, Journal of Statistical Physics, 53 (1988), pp. 1061–1072, https://doi.org/10.1007/BF01023857.

[50] H. Pichler, S.-T. Wang, L. Zhou, S. Choi, and M. D. Lukin, *Quantum optimization for maximum independent set using rydberg atom arrays*, 2018, https://doi.org/10.48550/ARXIV.1808.10816, https://arxiv.org/abs/1808.10816.

[51] M. Rahman and B. Virág, *Local algorithms for independent sets are half-optimal*, The Annals of Probability, 45 (2017), https://doi.org/10.1214/16-aop1094.

[52] A. Schönhage and V. Strassen, *Schnelle multiplikation grosser zahlen*, Computing, 7 (1971), pp. 281–292, https://doi.org/10.1007/BF02242355.

[53] Y. Shitov, *The complexity of tropical matrix factorization*, Advances in Mathematics, 254 (2014), pp. 138–156, https://doi.org/10.1016/j.aim.2013.12.013.

[54] A. A. Stepanov and D. E. Rose, *From mathematics to generic programming*, Pearson Education, 2014, https://www.fm2gp.com/.

[55] Q. Wu and J.-K. Hao, *A review on algorithms for maximum clique problems*, European Journal of Operational Research, 242 (2015), pp. 693–709, https://doi.org/10.1016/j.ejor.2014.09.064.

[56] Y.-Z. Xu, C. H. Yeung, H.-J. Zhou, and D. Saad, *Entropy inflection and invisible low-energy states: Defensive alliance example*, Physical Review Letters, 121 (2018), https://doi.org/10.1103/physrevlett.121.210602.

[57] C.-N. Yang and T.-D. Lee, *Statistical theory of equations of state and phase transitions. i. theory of condensation*, Physical Review, 87 (1952), p. 404, https://doi.org/10.1103/PhysRev.87.404.

Appendix A. An alternative way to construct the tensor network.

Let us characterize the independent set problem on graph $G = (V, E)$ as an energy model with two parts

$$(A.1) \qquad \mathcal{E}(G, s) = -\sum_{i \in V} w_i s_i + \infty \sum_{(i,j) \in E} s_i s_j$$

where s_i is a spin on vertex $i \in V$ and w_i is an onsite energy term associated with it. The first part corresponds to the negative independent set size and the second part describes the independence constraint, which corresponds to the Rydberg blockade [50, 19] in cold atom arrays or the repulsive force in hardcore lattice models [18, 21]. The partition function is defined as

$$
\begin{aligned}
(A.2) \qquad Z(G, \beta) &= \sum_s e^{-\beta \mathcal{E}(G,s)} = \sum_{s \in \mathcal{I}(G)} e^{\beta \sum w_i s_i} \\
&= \sum_{k=0}^{\alpha(G)} a(k) e^{\beta k} \qquad (k = \sum w_i s_i)
\end{aligned}
$$

where $\mathcal{I}(G)$ is the set of independent sets of graph G, $\alpha(G)$ is the absolute value of the minimum energy (maximum independent set size), $a(k)$ is the number of spin configurations with energy $-k$ (independent sets of size k). The partition function can be expressed as a tensor network by placing a vertex tensor on each spin $i \in V$

$$(A.3) \qquad W^{(i)} = \begin{pmatrix} 1 \\ e^{\beta w_i} \end{pmatrix},$$

and an edge tensor on each bond $(u, v) \in E$

$$(A.4) \qquad B^{(u,v)} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix},$$

where the 0 in the edge tensor comes from $e^{-\beta \infty}$ in the second term of Equation (A.1), which is the independence constraint. By letting $x = e^\beta$, we get the tensor network for computing

791   the independence polynomial as described by Equation (4.2) and Equation (4.3). If we further
792   let $w_i = 1$, the second line of Equation (A.2) is equivalent to the independence polynomial.

Appendix B. Hard problems and tensor networks.

794   **B.1. Maximal independent sets and maximal cliques.** In this section, we focus the
795   discussion on the maximal independent sets problem since finding maximal cliques of a graph
796   is equivalent to finding the maximal independent sets of its complement. Let $G = (V, E)$ be a
797   graph; we denote the neighborhood of a vertex $v \in V$ as $N(v)$. A maximal independent set I_m
798   is an independent set such that no $v \in V$ satisfies $I_m \cap (\{v\} \cup N[v]) = \emptyset$, i.e. an independent set
799   that cannot become a larger one by adding a new vertex. The corresponding tensor network
800    $\mathcal{N}_{\mathrm{mIS}}(G)$ can be specified as

$$
\begin{aligned}
\Lambda &= \{s_v \mid v \in V\}, \\
\mathcal{T} &= \{T^{(v)}_{s_{N(v,1)} s_{N(v,2)} \dots s_{N(v,d(v))} s_v} \mid v \in V\}, \\
\sigma_o &= \varepsilon,
\end{aligned}
\tag{B.1}
$$

802   where we defined a tensor for each $v \in V$ and its neighborhood $N(v)$ as

$$
T^{(v)}_{s_{N(v,1)} s_{N(v,2)} \dots s_{N(v,d(v))} s_v} =
\begin{cases}
s_v x_v^{w_v} & s_{N(v,1)} = s_{N(v,2)} = \dots = s_{N(v,d(v))} = 0, \\
1 - s_v & \text{otherwise.}
\end{cases}
\tag{B.2}
$$

804   Here, $N(v, k)$ is the kth vertex in $N(v)$ and $d(v) = |N(v)|$ is the degree of vertex v. If $s_v = 1$, then
805   none of its neighbours can be a member of I_m by the independence constraint, contributing a
806   factor $x_v^{w_v}$. If $s_v = 0$, then at least on of its neighbors must be in I_m by the maximal constraint,
807   contributing a unit factor. For a degree 2 vertex v, the tensor has the following form

$$
T^{(v)} =
\left(
\begin{pmatrix}
0 & 1 \\
1 & 1
\end{pmatrix}
\begin{pmatrix}
x_v^{w_v} & 0 \\
0 & 0
\end{pmatrix}
\right).
\tag{B.3}
$$

809

810   THEOREM B.1. *The tensor network representation of a maximal independent set problem*
811   *on a graph $G = (V, E)$ (Equation (B.1)) can be contracted in $cc(\mathcal{N}_{mIS}(G)) = O(|V|)2^{O(\mathrm{tw}(G)\Delta)}$*
812   *number of additions and multiplications, where Δ is the maximum degree of vertices in G.*

813   *Proof.* The prefactor $|V|$ comes from the number of tensors, while the contraction
814   complexity of pair-wise tensor contraction is closely related to the treewidth of the line
815   graph of its hypergraph representation $\mathrm{tw}(L(\mathcal{N}_{mIS}(G)))$. In the following, we will show this
816   quantity is upper bounded by $(\Delta + 1)$ times the treewidth of G. In the line graph $L(\mathcal{N}_{mIS}(G))$,
817   a tensor $T^{(v)}$ corresponds to a clique over $N(v) \cup \{v\}$. In the following, we will show given an
818   optimal tree decomposition of G, it is always possible to include all cliques into the bags by
819   increasing the bag size by a factor $(\Delta + 1)$. Let $v \in V$ be a vertex and $N(v)$ be its
820   neighborhood; we first arbitrarily pick an edge $u \in N(v)$, then by the definition of tree
821   decomposition, we can find a bag containing this edge, and lastly we include all $N(v) \cup \{v\}$
822   into this bag. Hence the maximum tensor rank during contraction is upper bounded by
823    $\mathrm{tw}(G)\Delta$, proving the theorem. $\square$

824   Let us consider the graph in Example 3. The corresponding tensor network structure for
825   computing the maximal independent polynomial has the following hypergraph
826   representation.
827
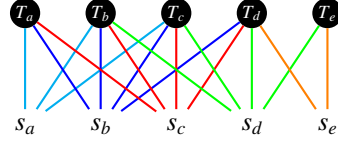
By contracting this tensor network with generic element types, we can compute the maximal independent set properties such as the maximal independence polynomial and the enumeration of maximal independent sets. The maximal independence polynomial is defined as

$$(B.4) \qquad D_m(G, x) = \sum_{k=0}^{\alpha(G)} b_k x^k,$$

where b_k is the number of maximal independent sets of size k. Comparing with the independence polynomial in Equation (5.1), we have $b_k \leq a_k$ and $b_{\alpha(G)} = a_{\alpha(G)}$. $D_m(G, 1)$ counts the total number of maximal independent sets [28, 43]; to our knowledge, the best algorithm has a time complexity $O(1.3642^{|V|})$ [28].

The benchmark of computing the maximal independent set properties on 3-regular graphs is shown in Section SM1.

B.2. Matching problem. A k-matching in a graph $G = (V, E)$ is a set of k edges that no two of which have a vertex in common. We map an edge $(u, v) \in E$ to a degree of freedom $\langle u, v \rangle \in \{0, 1\}$ in a tensor network, where 1 means an edge is in the set and 0 otherwise. The tensor network representation for the matching problem $\mathcal{N}_{\text{match}}$ can be specified as

$$(B.5) \qquad \begin{aligned} \Lambda &= \{\langle u, v \rangle \mid (u, v) \in E\}, \\ \mathcal{T} &= \{W^{(v)}_{\langle v, N(v,1) \rangle \langle v, N(v,2) \rangle \dots \langle v, N(v,d(v)) \rangle} \mid v \in V\} \cup \{B^{(u,v)}_{\langle u,v \rangle} \mid (u,v) \in E\}, \\ \sigma_o &= \varepsilon, \end{aligned}$$

where for each $v \in V$, we define a vertex tensor over its neighborhood $N(v)$ as

$$(B.6) \qquad W^{(v)}_{\langle v, N(v,1) \rangle \langle v, N(v,2) \rangle \dots \langle v, N(v,d(v)) \rangle} = \begin{cases} 1, & \sum_{i=1}^{d(v)} \langle v, N(v,i) \rangle \leq 1, \\ 0, & \text{otherwise,} \end{cases}$$

and for each bond $(u, v) \in E$, we define a rank one tensor as

$$(B.7) \qquad B^{(u,v)}_{\langle u,v \rangle} = \begin{cases} 1, & \langle u, v \rangle = 0 \\ x^{w_{\langle u,v \rangle}}_{\langle u,v \rangle}, & \langle u, v \rangle = 1 \end{cases}.$$

Here, $N(v, k)$ is the kth vertex in $N(v)$ and $d(v) = |N(v)|$ is the degree of vertex v; a label $\langle v, u \rangle$ is equivalent to $\langle u, v \rangle$. W tensor specifies the constraint that a vertex cannot be shared by two edges in the edge set, and an edge tensor carries the weights.

THEOREM B.2. *The tensor network representation of a matching problem on graph $G = (V, E)$ (Equation (B.5)) can be contracted in $cc(\mathcal{N}_{match}(G)) = O(|V|)2^{O(\text{tw}(L(G)))}$ number of additions and multiplications, where $L(G)$ is the line graph of G.*

Proof. To contract the tensor network, we first absorb edges tensors into the vertex tensors, which does not increase computational complexity. After this, the resulting tensor network is isomorphic to G. Hence the contraction complexity is $O(|V|)2^{O(\text{tw}(L(G)))}$. ☐

857 Let $x_{\langle u,v\rangle}^{w_{\langle u,v\rangle}} = x$; the tensor network contraction corresponds to the matching polynomial

858 (B.8)
$$M(G, x) = \sum_{k=1}^{|V|/2} c_k x^k,$$

859 where k is the size of an edge set, and a coefficient c_k is the number of k-matchings.

860 **B.3. Vertex coloring.** Let $G = (V, E)$ be a graph. A vertex coloring is an assignment
861 of colors to each vertex $v \in V$ such that no edge connects two identically colored vertices.
862 In a k-coloring problem, the number of colors is limited to less or equal to k. Let us use the
863 3-coloring problem as an example to show how to reduce it to tensor contractions. We first
864 map a vertex $v \in V$ to a degree of freedom $c_v \in \{0, 1, 2\}$. The tensor network for the vertex
865 coloring problem $\mathcal{N}_{3\text{-color}}(G)$ can be specified as

866 (B.9)
$$\Lambda = \{c_v \mid v \in V\},$$
$$\mathcal{T} = \{W_{c_v}^{(v)} \mid v \in V\} \cup \{B_{c_u c_v}^{(u,v)} \mid (u, v) \in E\},$$
$$\sigma_o = \varepsilon,$$

867 where for each vertex $v \in V$, we define a tensor labelled by c_v

868 (B.10)
$$W^{(v)} = \begin{pmatrix} 1_{c_v=r} \\ 1_{c_v=g} \\ 1_{c_v=b} \end{pmatrix},$$

869 and for each edge $(u, v) \in E$, we define a tensor labelled by (c_u, c_v) as

870 (B.11)
$$B^{(u,v)} = \begin{pmatrix} 0 & x^{w_{uv}} & x^{w_{uv}} \\ x^{w_{uv}} & 0 & x^{w_{uv}} \\ x^{w_{uv}} & x^{w_{uv}} & 0 \end{pmatrix},$$

871 where subscripts $c_v = r$, $c_v = g$ and $c_v = b$ are for labeling the color configurations. B tensors
872 are for specifying the coloring constraints and W tensors are for labeling the solutions.

873   THEOREM B.3. *The tensor network representation of a K-coloring problem on a graph*
874 $G = (V, E)$ *(Equation* (B.9)*) can be contracted in* $cc(\mathcal{N}_{K\text{-color}}(G)) = O(|E|)K^{O(\text{tw}(G))}$ *number of*
875 *additions and multiplications.*

876 The proof is similar to that for Theorem 4.1 except the dimension of each degree of freedom
877 is K. Let $x^{w_{uv}} = x$ and $r_v = g_v = b_v = 1$; we then have a graph polynomial, in which the k-th
878 coefficient is the number of coloring that k bonds satisfy constraints. If a graph is colorable,
879 the maximum order of this polynomial should be equal to the number of edges in this graph.
880 Similarly, one can define an edge coloring problem by defining the tensor network on the line
881 graph of G.

882   **B.4. Cutting problem.** In graph theory, a cut is a partition of the vertices of a graph
883 into two disjoint subsets, which is also known as the spin-glass problem in statistical physics.
884 Let $G = (V, E)$ be a graph. We associate a weight w_v to each $v \in V$. To reduce the cutting
885 problem on G to the contraction of a tensor network, we first define a Boolean degree of
886 freedom $s_v \in \{0, 1\}$ for each vertex $v \in V$. The tensor network representation for the cutting
887 problem $\mathcal{N}_{\text{cut}}$ can be specified as

888 (B.12)
$$\mathcal{T} = \{B_{s_u s_v}^{(u,v)} \mid (u, v) \in E\},$$
$$\Lambda = \{s_v \mid v \in V\},$$
$$\sigma_o = \varepsilon,$$

25

where for each edge $(u, v) \in E$, we define an edge matrix labelled by (s_u, s_v) as

$$(\text{B.13}) \qquad B^{(u,v)} = \begin{pmatrix} 1 & x_v^{w_{uv}} \\ x_u^{w_{uv}} & 1 \end{pmatrix}.$$

Here, variables $x_u^{w_{uv}}$ and $x_v^{w_{uv}}$ are for a cut on this edge or a domain wall in a spin glass problem.

THEOREM B.4. *The tensor network representation of a cuting problem on a graph $G = (V, E)$ (Equation (B.12)) can be contracted in $cc(\mathcal{N}_{cut}(G)) = O(|E|)2^{O(\text{tw}(G))}$ number of additions and multiplications.*

The proof is similar to that for Theorem 4.1. Let $x_u^{w_{uv}} = x_v^{w_{uv}} = x$; we have a graph polynomial similar to the previous ones, in which the kth coefficient is two times the number of cut configurations that have size k (i.e. cutting k edges).

B.5. Dominating Set. In graph theory, a dominating set for a graph $G = (V, E)$ is a subset $D \subseteq V$ such that every vertex not in D is adjacent to at least one member of D. To reduce this problem to the contraction of a tensor network, we first map a vertex $v \in V$ to a Boolean degree of freedom $s_v \in \{0, 1\}$. The tensor network for the dominating set problem $\mathcal{N}_{\text{dom}}$ can be specified as

$$(\text{B.14}) \qquad \begin{aligned} \Lambda &= \{s_v \mid v \in V\}, \\ \mathcal{T} &= \{T^{(v)}_{s_{N(v,1)} s_{N(v,2)} \cdots s_{N(v,d(v))} s_v} \mid v \in V\}, \\ \sigma_o &= \varepsilon, \end{aligned}$$

where for each vertex v, we define a tensor on its closed neighborhood $\{v\} \cup N(v)$ as

$$(\text{B.15}) \qquad T^{(v)}_{s_{N(v,1)} s_{N(v,2)} \cdots s_{N(v,d(v))} s_v} = \begin{cases} 0 & s_{N(v,1)} = s_{N(v,2)} = \ldots = s_{N(v,d(v))} = s_v = 0, \\ 1 & s_v = 0, \\ x_v^{w_v} & \text{otherwise.} \end{cases}$$

Here, w_v is the weight associated with the vertex v, $N(v, k)$ is the kth vertex in $N(v)$ and $d(v) = |N(v)|$ is the degree of vertex v. This tensor implies a configuration having a closed neighborhood of v not in D ($s_{N(v,1)} = s_{N(v,2)} = \ldots = s_{N(v,d(v))} = s_v = 0$) cannot be a dominating set. Otherwise, if v is in D, this tensor contributes a multiplicative factor $x_v^{w_v}$ to the output.

THEOREM B.5. *The tensor network representation of a dominating set problem on a graph $G = (V, E)$ (Equation (B.14)) can be contracted in $cc(\mathcal{N}_{dom}(G)) = O(|V|)2^{O(\text{tw}(G)\Delta)}$ number of additions and multiplications.*

The proof is similar to that for Theorem B.1. The graph polynomial for the dominating set problem is known as the domination polynomial [3]

$$(\text{B.16}) \qquad D(G, x) = \sum_{k=0}^{\gamma(G)} d_k x^k,$$

where d_k is the number of dominating sets of size k.

B.6. Boolean satisfiability Problem. The Boolean satisfiability problem is the problem of determining if there exists an assignment that satisfies a given Boolean formula. One can specify a satisfiable problem in the conjunctive normal form (CNF), i.e. a conjunction of clauses (or disjunctions of Boolean literals). Given the alphabet of Boolean variables V and its negation $\neg V = \{\neg v \mid v \in V\}$, a CNF can be formally defined as

$$(\text{B.17}) \qquad \text{CNF} = \bigwedge_{k=1}^{M} \bigvee_{i=1}^{|C^{(k)}|} C_i^{(k)}$$

26

where M is the number of clauses, $C^{(k)}$ is the kth clause and $C^{(k)}_i \in V \cup \neg V$ is the ith literal in it. The standard tensor network can be used to study the counting version of the satisfiability problem [9], while in the following, we will show a generic reduction from the problem of solving a CNF to a tensor network contraction for solving more solution space properties. We first map each Boolean literal $v \in V$ to a Boolean degree of freedom $s_v \in \{0, 1\}$. $s_v = 0$ stands for variable v having value `false` while $s_a = 1$ stands for having value `true`. The tensor network $\mathcal{N}_{\mathrm{CNF}}$ can be specified as

$$\text{(B.18)} \quad \begin{aligned} \Lambda &= \{s_v \mid v \in V\}, \\ \mathcal{T} &= \{W^{(v)}_v \mid v \in V\} \cup \{T^{(k)}_{s_{N(k,1)} s_{N(k,2)} \cdots s_{N(k,d(k))}} \mid k = 1, \ldots, M\}, \\ \sigma_o &= \varepsilon, \end{aligned}$$

where a tensor defined on literal $v \in V$ is

$$\text{(B.19)} \quad W^{(v)} = \begin{pmatrix} 1 \\ 1_v \end{pmatrix}$$

and a tensor defined on the clause $C^{(k)}$ is

$$\text{(B.20)} \quad T^{(k)}_{s_{N(k,1)} s_{N(k,2)} \cdots s_{N(k,d(k))}} = \begin{cases} x^{w_k}, & C^{(k)} \text{ is satisfied by } (s_{N(k,1)}, s_{N(k,2)}, \ldots, s_{N(k,d(k))}), \\ 1, & \text{otherwise} \end{cases},$$

where $N(k, i) = \begin{cases} \neg C^{(k)}_i, & C^{(k)}_i \in \neg V \\ C^{(k)}_i, & C^{(k)}_i \in V \end{cases}$ is the ith literal in $C^{(k)}$ with its negation sign removed and $d(k)$ is the number of boolean variables in it; w_k is the weight associated with clause $C^{(k)}$.

THEOREM B.6. *The tensor network representation of a CNF (Equation (B.18)) can be contracted in $cc(\mathcal{N}_{CNF}) = O(M)2^{O(\mathrm{tw}(H))}$ number of additions and multiplications, where M is the number of clauses, H is a hypergraph constructed by mapping a variable $v \in V$ to a vertex and the kth clause $C^{(k)}$ to a hyperedge connecting $\{N(k, i) \mid i = 1, \ldots, d(k)\}$.*

This can be proved by showing the hypergraph H is the line graph of $\mathcal{N}_{\mathrm{CNF}}$. Let $x^{w_k} = x$ and $1_v = 1$; one can get a polynomial, in which the k-th coefficient gives the number of assignments that k clauses are satisfied.

B.7. Set packing. Suppose one has a finite set V and a list of subsets of V, denoted as S. Then, the set packing problem asks if some k subsets in S are pairwise disjoint. It is the hypergraph generalization of the independent set problem, where a set corresponds to a vertex and an element corresponds to a hyperedge. The generic tensor network for the set packing problem $\mathcal{N}_{\mathrm{pack}}$ also has a similar form as that for the independent set problem

$$\text{(B.21)} \quad \begin{aligned} \Lambda &= \{s_\sigma \mid \sigma \in S\}, \\ \mathcal{T} &= \{B^{(v)}_{s_{N(v,1)} \cdots s_{N(v,d(v))}} \mid v \in V\} \cup \{W^{(\sigma)}_{s_\sigma} \mid \sigma \in S\}, \\ \sigma_o &= \varepsilon, \end{aligned}$$

where for each $v \in V$, we have the constraints over sets, $N(v) = \{\sigma \mid \sigma \in S \wedge v \in \sigma\}$, that containing it as

$$\text{(B.22)} \quad B^{(v)}_{N(v,1)N(v,2)\ldots N(v,d(v))} = \begin{cases} 1, & s_{N(v,1)} + s_{N(v,2)} + \ldots + s_{N(v,d(v))} \leq 1, \\ 0, & \text{otherwise}. \end{cases}$$

27

and the vertex tensor for each $\sigma \in S$

(B.23)
$$W^{(\sigma)} = \begin{pmatrix} 1 \\ x_\sigma^{w_\sigma} \end{pmatrix},$$

where $N(v, k)$ is the kth element in $N(v)$ and $d(v) = |N(v)|$ is the number of elements in $N(v)$.

THEOREM B.7. *The tensor network representation of a set packing problem (Equation (B.21)) can be contracted in $cc(\mathcal{N}_{pack}) = O(|S|)2^{O(\text{tw}(H))}$ number of additions and multiplications, where $|S|$ is the number of sets, H is a hypergraph constructed by mapping a set $\sigma \in S$ to a vertex and element $v \in V$ to a hyperedge connecting sets, $N(v)$, that containing it.*

This can be proved by showing the hypergraph H is the line graph of $\mathcal{N}_{pack}$.

B.8. Set covering. Suppose one has a finite set V and a list of subsets of V, denoted as S. The set covering problem aims to find the minimum number of sets in S that incorporate (cover) all of elements in V. To get the generic tensor network representation, we first map a set $\sigma \in S$ to a Boolean degree of freedom $s_\sigma \in \{0, 1\}$. Then the tensor network representation for the set covering problem $\mathcal{N}_{cover}$ can be specified as

(B.24)
$$\begin{aligned} \Lambda &= \{s_\sigma \mid \sigma \in \mathcal{S}\}, \\ \mathcal{T} &= \{W^{(\sigma)}_{s_\sigma} \mid \sigma \in S\} \cup \{B^{(v)}_{s_{N(v,1)} s_{N(v,2)} \cdots s_{N(\sigma,d(v))}} \mid v \in V\}, \\ \sigma_o &= \varepsilon, \end{aligned}$$

where for each s_σ, we define a parameterized rank-one tensor indexed by it as

(B.25)
$$W^{(\sigma)} = \begin{pmatrix} 1 \\ x_\sigma^{w_\sigma} \end{pmatrix}$$

where x_σ is a generic typed variable associated with σ and w_σ is a positive integer as the weight. For each element $v \in V$, we can define a constraint over all $s \in S$ containing this element, i.e. $N(v) = \{s \mid s \in S \wedge v \in s\}$, as

(B.26)
$$B^{(v)}_{s_{N(v,1)} s_{N(v,2)} \cdots s_{N(v,d(v))}} = \begin{cases} 0, & s_{N(v,1)} = s_{N(v,2)} = \ldots = s_{N(v,d(v))} = 0, \\ 1, & \text{otherwise.} \end{cases}$$

where $N(v, k)$ is the kth element in $N(v)$ and $d(v) = |N(v)|$ is the number of elements in $N(v)$. If a subset of S does not include any sets containing element v, then the corresponding entry is zero.

THEOREM B.8. *The tensor network representation of a set covering problem (Equation (B.24)) can be contracted in $cc(\mathcal{N}_{cover}) = O(|S|)2^{O(\text{tw}(H))}$ number of additions and multiplications, where $|S|$ is the number of sets, H is a hypergraph constructed by mapping a set $\sigma \in S$ to a vertex and element $v \in V$ to a hyperedge connecting sets $N(v)$, that containing it.*

This can be proved by showing the hypergraph H is the line graph of $\mathcal{N}_{cover}$.

Appendix C. Bounding the MIS enumeration space.

When using the algebra in Equation (7.3: P1+SN) to enumerate all MISs, the program often stores significantly more intermediate configurations than necessary. To reduce the space overhead, we will show how to bound the searching space using the MIS size $\alpha(G)$. The bounded contraction consists of three stages as shown in Figure 6. (a) We first compute
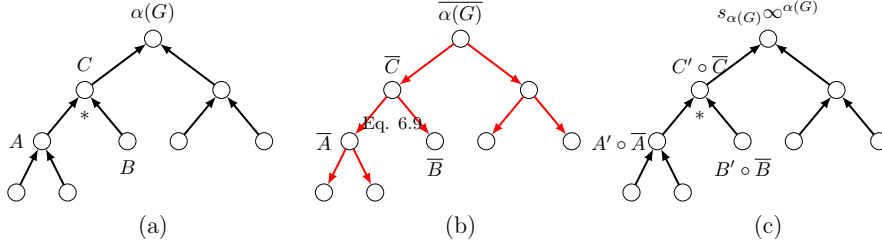
28

Figure 6: Bounded enumeration of maximum independent sets. Here, a circle is a tensor, an arrow specifies the execution direction of a function, $\overline{A}$ is the Boolean mask for A and $\circ$ is the Hadamard (element-wise) multiplication. (a) is the forward pass with tropical algebra (Equation (6.4: T)) for computing $\alpha(G)$. (b) is the backward pass for computing Boolean gradient masks. (c) is the masked tensor network contraction with tropical algebra combined with sets (Equation (7.3: P1+SN)) for enumerating configurations.

the value of $\alpha(G)$ with tropical algebra and cache all intermediate tensors. (b) Then, we compute a Boolean mask for each cached tensor, where we use a Boolean `true` to represent a tensor element having a contribution to the MIS and Boolean `false` otherwise. (c) Finally, we perform masked tensor network contraction (i.e. discarding elements masked `false`) using the element type with the algebra in Equation (7.3: P1+SN) to obtain all MIS configurations. The crucial part is computing the masks in step (b). Note that these masks correspond to tensor elements with non-zero gradients to the MIS size; we can compute these masks by back-propagating the gradients. To derive the back-propagation rule for tropical tensor contraction, we first reduce the problem to finding the back-propagation rule of a tropical matrix multiplication $C = AB$. Since $O_{ik} = \bigoplus_j A_{ij} \odot B_{jk} = \max_j A_{ij} \odot B_{jk}$ with tropical algebra, we have the following inequality

(C.1)
$$A_{ij} \odot B_{jk} \le C_{ik}.$$

Here $\le$ on tropical numbers are the same as the real-number algebra. The equality holds for some j', which means $A_{ij'}$ and $B_{j'k}$ have contributions to C_{ik}. Intuitively, one can use this relation to identify elements with nonzero gradients in A and B, but if doing this directly, one loses the advantage of using BLAS libraries [2] for high performance. Since $A_{ij} \odot B_{jk} = A_{ij} + B_{jk}$, one can move B_{jk} to the right hand side of the inequality:

(C.2)
$$A_{ij} \le C_{ik} \odot B_{jk}^{\circ-1}$$

where $\circ-1$ is the element-wise multiplicative inverse on tropical algebra (which is the additive inverse on real numbers). The inequality still holds if we take the minimum over k:
(C.3)

$$A_{ij} \le \min_k (C_{ik} \odot B_{jk}^{\circ-1}) = \left(\max_k \left(C_{ik}^{\circ-1} \odot B_{jk} \right) \right)^{\circ-1} = \left(\bigoplus_k \left(C_{ik}^{\circ-1} \odot B_{jk} \right) \right)^{\circ-1} = \left(C^{\circ-1} B^{\mathsf{T}} \right)_{ij}^{\circ-1}.$$

On the right-hand side, we transform the operation into a tropical matrix multiplication so that we can utilize the fast tropical BLAS routines [2]. Again, the equality holds if and only if the element A_{ij} has a contribution to C (i.e. having a non-zero gradient). Let the gradient mask for C be $\overline{C}$; the back-propagation rule for gradient masks reads

(C.4)
$$\overline{A}_{ij} = \delta \left(A_{ij}, \left(\left(C^{\circ-1} \circ \overline{C} \right) B^{\mathsf{T}} \right)_{ij}^{\circ-1} \right),$$

29

where δ is the Dirac delta function that returns one if two arguments have the same value and zero otherwise, $\circ$ is the element-wise product, Boolean false is treated as the tropical number $\mathbb{0}$, and Boolean true is treated as the tropical number $\mathbb{1}$. This rule defined on matrix multiplication can be easily generalized to tensor contraction by replacing the matrix multiplication between $C^{\circ-1} \circ \overline{C}$ and B^{T} by a tensor contraction. With the above method, one can significantly reduce the space needed to store the intermediate configurations by setting the tensor elements masked false to zero during contraction.

Appendix D. The fitting approach to computing the independence polynomial. In this section, we propose to find the independence polynomial by fitting $\alpha(G)+1$ random pairs of x_i and $y_i = I(G, x_i)$. One can then compute the independence polynomial coefficients a_i by solving the linear equation:

(D.1)
$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{\alpha(G)} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{\alpha(G)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{\alpha(G)} & x_{\alpha(G)}^2 & \cdots & x_{\alpha(G)}^{\alpha(G)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{\alpha(G)} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\alpha(G)} \end{pmatrix}.$$

Unlike using the polynomial numbers in Equation (5.2: PN), the fitting approach does not have the linear overhead in space. However, since the independence polynomial coefficients can have a huge order-of-magnitude range, the round-off errors can be larger than the value itself when using floating-point numbers in computation. To avoid using the arbitrary precision number that can be very slow and is incompatible with GPU devices, we introduce the following finite-field algebra GF(p) approach:

(D.2: GF(p))
$$\begin{aligned} x \oplus y &= x + y \quad (\bmod\ p), \\ x \odot y &= xy \quad (\bmod\ p), \\ \mathbb{0} &= 0, \\ \mathbb{1} &= 1. \end{aligned}$$

Regarding the finite-field algebra, we have the following observations:

1. One can use Gaussian elimination [29] to solve the linear equation Equation (D.1) since it is a generic algorithm that works for any elements with field algebra. The multiplicative inverse of a finite-field algebra can be computed with the extended Euclidean algorithm.
2. Given the remainders of a larger unknown integer x over a set of co-prime integers $\{p_1, p_2, \ldots, p_n\}$, x (mod $p_1 \times p_2 \times \ldots \times p_n$) can be computed using the Chinese remainder theorem. With this, one can infer big integers from small integers.

With these observations, we develop Algorithm D.1 to compute the independence polynomial exactly without introducing space overheads. The algorithm iterates over a sequence of large prime numbers until convergence. In each iteration, we choose a large prime number p, and contract the tensor networks to evaluate the polynomial for each variable $\chi = (x_0, x_1, \ldots, x_{\alpha(G)})$ on GF(p) and denote the outputs as $(y_0, y_1, \ldots, y_{\alpha(G)})$ (mod p). Then we solve Equation (D.1) using Gaussian elimination on GF(p) to find the coefficient modulo p, $A_p \equiv (a_0, a_1, \ldots, a_{\alpha(G)})$ (mod p). As the last step of each iteration, we apply the Chinese remainder theorem to update A (mod P) to A (mod $P \times p$), where P is a product of all prime numbers chosen in previous iterations. If this number does not change compared with the previous iteration, it indicates the convergence of the result and the program terminates. All computations are done with integers of fixed-width W except the last step of applying the Chinese remainder theorem, where we use arbitrary precision integers to represent the counting.

30

Algorithm D.1 Computing the independence polynomial exactly without integer overflow

---

Let $P = 1$, W be the integer width, vector $\chi = (0, 1, 2, \ldots, \alpha(G))$, matrix $X_{ij} = (\chi_i)^j$, where $i, j = 0, 1, \ldots, \alpha(G)$

while *true* **do**

    compute the largest prime p that $\gcd(p, P) = 1$ and $p < 2^W$

    for $i = 0 \ldots \alpha(G)$ **do**

        $y_i \pmod{p} = \text{contract\_tensor\_network}(\chi_i \pmod{p})$; // on GF(p)

    end

    $A_p = (a_0, a_1, \ldots, a_{\alpha(G)}) \pmod{p} = \text{gaussian\_elimination}(X, (y_0, y_1, \ldots, y_{\alpha(G)}) \pmod{p})$

    $A_{P \times p} = \text{chinese\_remainder}(A_P, A_p)$

    if $A_P = A_{P \times p}$ **then**

        return A_P ; // converged

    end

    $P = P \times p$

end

1054   Alternatively, one can use a faster but less accurate Fourier transformation based method
1055 to fit this polynomial, which is detailed and benchmarked in Section SM1.

1056   **Appendix E. Integer sequence formed by the number of independent sets.** We
1057 computed the number of independent sets on square lattices and King's graphs with our
1058 generic tensor network contraction algorithm on GPUs. The tensor element type is the
1059 finite-field algebra so that we can reach an arbitrary precision. We also computed the
1060 independence polynomial for these lattices up to size 30×30 in our GitHub repository.

Table 2: The number of independent sets for square lattice graphs of size $L \times L$. This forms the integer sequence OEIS A006506. Here we only include two updated entries for $L = 38, 39$, which, to our knowledge, has not been computed before [14].

L	square lattice graphs
38	616 412 251 028 728 207 385 738 562 656 236 093 713 609 747 387 533 907 560 081 990 229 746 115 948 572 583 817 557 035 128 726 922 565 913 748 716 778 414 190 432 479 964 245 067 083 441 583 742 870 993 696 157 129 887 194 203 643 048 435 362 875 885 498 554 979 326 352 127 528 330 481 118 313 702 375 541 902 300 956 879 563 063 343 972 979
39	29 855 612 447 544 274 159 031 389 813 027 239 335 497 014 990 491 494 036 487 199 167 155 042 005 286 230 480 609 472 592 158 583 920 411 213 748 368 073 011 775 053 878 033 685 239 323 444 700 725 664 632 236 525 923 258 394 737 964 155 747 730 125 966 370 906 864 022 395 459 136 352 378 231 301 643 917 282 836 792 261 715 266 731 741 625 623 207 330 411 607