Toward Minimum Graphic Parity Networks

Yixin Cao[2], Yiren Lu[1,3], Junhong Nie[1,3], Xiaoming Sun[1,3] and Guojing Tian[1,3]

[1]*State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China*
[2]*Department of Computing, Hong Kong Polytechnic University, Hong Kong, China*
[3]*School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China*

Abstract

Quantum circuits composed of CNOT and R_z are fundamental building blocks of many quantum algorithms, so optimizing the synthesis of such quantum circuits is crucial. We address this problem from a theoretical perspective by studying the graphic parity network synthesis problem. A graphic parity network for a graph G is a quantum circuit composed solely of CNOT gates where each edge of G is represented in the circuit, and the final state of the wires matches the original input. We aim to synthesize graphic parity networks with the minimum number of gates, specifically for quantum algorithms addressing combinatorial optimization problems with Ising formulations. We demonstrate that a graphic parity network for a connected graph with n vertices and m edges requires at least $m + n - 1$ gates. This lower bound can be improved to $m + \Omega(m) = m + \Omega(n^{1.5})$ when the shortest cycle in the graph has a length of at least five. We complement this result with a simple randomized algorithm that synthesizes a graphic parity network with expected $m + O(n^{1.5}\sqrt{\log n})$ gates.

Additionally, we begin exploring connected graphs that allow for graphic parity networks with exactly $m + n - 1$ gates. We conjecture that all such graphs belong to a newly defined graph class. Furthermore, we present a linear-time algorithm for synthesizing minimum graphic parity networks for graphs within this class. However, this graph class is not closed under taking induced subgraphs, and we show that recognizing it is NP-complete, which is complemented with a fixed-parameter tractable algorithm parameterized by the treewidth.

1 Introduction

Over the past decade, there has been significant progress in quantum computation, with advancements in both experimental and theoretical aspects [5, 17, 25, 7]. Compared to classical circuits, quantum circuits are more sensitive to noise, which is one of the bottlenecks limiting the scalability of quantum computers. Each quantum gate in a circuit may introduce some noise to the output quantum state, and as the circuit size increases, the accumulated noise can overwhelm any meaningful computational results. Since we are still in the noisy intermediate-scale quantum (NISQ) era, there exists a fundamental necessity to minimize the number of gates used in quantum circuits [23, 32, 15, 34], especially for two-qubit gates, which are even more vulnerable to noise than single-qubit gates [7, 20].

Subcircuits consisting only of CNOT and R_z gates appear in many quantum algorithms, such as quantum simulation [24], quantum approximate optimization algorithm (QAOA) [10], variational quantum eigensolver (VQE) [28], etc. To optimize such quantum circuits with massive $\{\text{CNOT}, R_z\}$-only components, a widely employed method is to independently resynthesize these components, and this problem has been extensively treated by *parity network synthesis* [1, 12]. Parity network is a CNOT-only quantum circuit. In a sense, CNOT gates can be viewed as the

quantum analog of classical XOR gates.[1] A CNOT gate applied to qubit wires i and j, denoted by $\textsc{cnot}(i,j)$, changes the value of wire j to $a \oplus b$, where $a, b \in \mathbb{F}_2$ are the values of wires i and j, respectively. Here, wire i is the *control wire*, and wire j is the *target wire*. A CNOT circuit is called a *parity network* for a set family S if every element of S appears in the circuit as a *term* at some intermediate point, and the final values of the wires are the same as the original input, denoted by $x_1, x_2, \ldots, x_n \in \mathbb{F}_2$ [1], and we say that a wire has the *term* $A = \{a_1, \ldots, a_{|A|}\}$ on it if the value of a wire evaluates to $x_{a_1} \oplus \cdots \oplus x_{a_{|A|}}$.

The set family S defines a hypergraph. In certain applications, it is actually a graph, namely all terms in S are two-element sets. For example, the maximum cut problem asks for a bipartition of the vertex set of a graph such that the number of edges between the two parts is maximized. The well-known quantum adiabatic algorithm (QAA) [11] and QAOA [10] use the following formulation to solve the maximum cut problem on a graph G:

$$\max_{x \in \{-1,1\}^{|V(G)|}} \sum_{(u,v) \in E(G)} \frac{1 - x_u x_v}{2}.$$

A principal component of the quantum circuit of these algorithms is a parity network with each term being a two-element set. A similar approach has been taken by many authors [33, 35, 36, 37, 4, 3, 2, 30], and there have been preliminary physical demonstrations of these quantum algorithms on NISQ devices [27, 14]. Recently, heuristic attempts have been made to synthesize small parity networks [1, 12, 9]. As far as we know, however, none of them is accompanied with analysis of performance. Such parity networks where all terms in S are two-element sets are called *graphic parity networks*, and see fig. 1 as examples. This paper is mainly concerned with the graphic parity network synthesis problem.

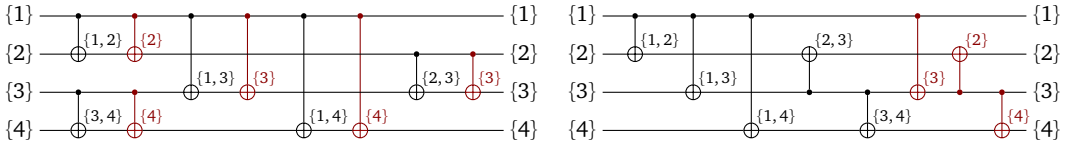


Figure 1: Two graphic parity networks for the graph with edges $\{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{3,4\}\}$, shown in fig. 2a. Each line represents a qubit wire labeled at the left. We use $i \bullet\!\!-\!\!\oplus\, j$ to denote $\textsc{cnot}(i,j)$. Indicated at the above right corner of $\oplus$ is the resulting term of this operation. The black terms correspond to the sets, and the red do not.

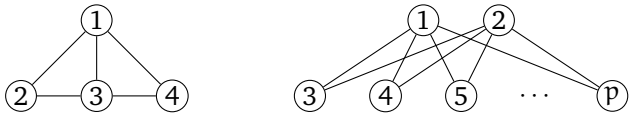


Figure 2: (a) A chordal graph, and (b) $K_{2,p}$, which can be made chordal by adding the edge $(1,2)$.

Our contributions. The number of gates in a graphic parity network is called its *size*. Let G be the input graph, and let n and m denote the numbers of vertices and edges, respectively,

[1] To make our work accessible to a general audience, we will minimize the use of quantum computing jargon in the main text. Consequently, our definitions and explanations might seem imprecise to quantum experts. For a more detailed and rigorous treatment, please refer to appendix A.

in G. A trivial graphic parity network is in the fashion of Figure 1a; i.e., generating an edge and cleaning it up immediately. This leads to a trivial upper bound of $2m$, which is tight, as witnessed by graphs in which each component consists of one or two vertices. On the other hand, the definition implies a trivial lower bound, m, which, although tight, is uninteresting because it can only hold on edgeless graphs.

Our first result is a nontrivial lower bound of the size of graphic parity networks.

Theorem 1.1. *A graphic parity network for a graph* G *contains at least* $m + n - c$ *gates, where* c *is the number of components of* G.

Again, this bound is tight: it matches the upper bound $2m$ for all forests. We say that a graphic parity network is *perfect* if its size is precisely $m + n - c$. A graph admits a graphic parity network that is perfect or close to perfect only when there are a lot of operations from two edge terms to another edge term, e.g., $\{1, 3\} \oplus \{1, 2\} = \{2, 3\}$ in Figure 1b. The element 1 is canceled from the target wire of this operation, which is hence called a *cancellation*. A cancellation can only happen when the three edges form a triangle. As we will see, for *chordal graphs*—graphs in which every induced cycle is a triangle,—it is easy to synthesize perfect graphic parity networks. This observation can be extended to graphs that can be turned into chordal graphs by adding a small number of edges, which contains many cycles of length three or four. See Figure 2 for examples. On the other hand, if the length of shortest cycles in a graph is five or more, we cannot do significantly better than the trivial construction of $2m$ gates.

Theorem 1.2. *For any positive integer* n*, there exists a graph* G *with* $\Omega\left(n\sqrt{n}\right)$ *edges such that the size of its minimum graphic parity networks is* $m + \Omega(m)$.

Motivated by these observations, we propose a randomized algorithm for synthesizing graphic parity networks. It processes the vertices in a random order, and for each vertex v, generates all the edges between v and latter vertices. The algorithm tries to minimize the size by exploiting the triangles and squares in the input graph. The expected size of the synthesized circuit almost matches the bound in Theorem 1.2. In particular, it produces very good results for dense graphs.

Theorem 1.3. *There is a polynomial-time algorithm synthesizing a graphic parity network with expected size* $m + O\left(n^{1.5}\sqrt{\log n}\right)$. *Moreover, if all but a constant number of vertices have degrees* $\Omega(n)$*, the synthesized graphic parity network has expected size* $m + O(n \log n)$.

A natural question is to characterize graphs admitting PERfect graphic PArity NEtworks, which we call *perpane*. An immediate consequence of Theorem 1.1 is that in a perfect graphic parity network, there always exists a wire that is not a target of any operation. We noticed that for all perpane graphs we have discovered, we can synthesize a perfect graphic parity network in which there exists a wire that is not a control of any operation, e.g., wires 2 and 4 in Figure 1b. All edges involving the vertex v corresponding to this wire have to be generated along it, and the removal of this wire leads to a reduced graphic parity network for the subgraph $G - v$. Thus, there must be precisely $d + 1$ operations targeting this wire, where d is the number of neighbors of v in G. Except for the first and the last, each operation on this wire makes a new edge term by cancellation. This motivates us to define *perfect cancellation orderings* and *perfect cancellation graphs*. The formal definition is technical and hence deferred to Section 2.2. All chordal graphs are perfect cancellation graphs: all perfect elimination orderings [29] are perfect cancellation orderings, but not the other way. As we will see, a perfect cancellation ordering can guide us in synthesizing a perfect graphic parity network for G in linear time.

Theorem 1.4. *All perfect cancellation graphs are perpane. Given a perfect cancellation graph* G *and a perfect cancellation ordering of* G*, we can synthesize a perfect graphic parity network for* G *in linear time.*

We conjecture that the two graph classes are equivalent: a graph is perpane if and only if it is a perfect cancellation graph. We leave it to the reader to verify that a simple cycle on four vertices is not a perfect cancellation graph, while it becomes one after adding a universal vertex, i.e., a vertex that is adjacent to all the vertices on the cycle. Thus, the class of perfect cancellation graphs is not *hereditary*, i.e., closed under taking induced subgraphs. The same holds for perpane graphs. This suggests that both classes are not easy to handle algorithmically. Indeed, finding a perfect cancellation ordering is computationally hard.

Theorem 1.5. *It is* NP*-complete to decide whether a graph is a perfect cancellation graph.*

Finally, we present a fixed-parameter tractable algorithm for recognizing perfect cancellation graphs, using the the treewidth of the input graph as the parameter. Similar to most algorithms using a tree decomposition, we use dynamic programming bottom-up. The main challenge is that a subgraph may need vertices from without to make a good ordering. As said, the class of perfect cancellation graphs is not hereditary.

Theorem 1.6. *The recognition of perfect cancellation graphs is fixed-parameter tractable parameterized by the treewidth of the input graph.*

Other related work. Two other important measures for quantum circuits or reversible circuits are the depth and the number of ancillae. The *depth* of a quantum circuit is the count of time steps needed to execute all the gates in the circuit in parallel; e.g., the first two gates in Figure 1a contribute one to the depth. An *ancilla bit* is a qubit whose input is the particular state $\{0\}$ and can be utilized as auxiliary space throughout the computation but must be recovered to $\{0\}$ at the end of computation. The circuit depth characterizes the running time of a quantum circuit, while the number of ancillae characterizes the extra space required by a quantum circuit. Any n-qubit CNOT circuit can be represented by an invertible matrix $\mathbb{M} \in \mathbb{F}_2^{n\times n}$, and the synthesis of CNOT circuit is equivalent to transforming $\mathbb{M}$ to identity by Gaussian elimination; see more details in the appendix. The main trick in minimizing circuit depth is to employ more ancillary qubits to eliminate multi-columns rather than one-column simultaneously [15, 22, 13]. Interestingly, this ultimately reduced to the parallel Gaussian elimination, which is inherently related to chordal graphs, also known as perfect elimination graphs. It is easy to see that any parity network for a connected graph has depth $\Omega(\log n)$. In fact, there is a trivial method synthesizing parity network for any graph in depth $O(\log n)$, as long as enough ancillae are given. This line of work is orthogonal to ours because it usually increases the size.

2 Graphic parity networks and perpane graphs

All graphs discussed in this paper are finite and simple. The vertex set and edge set of graph G are denoted by, respectively, $V(G)$ and $E(G)$. Throughout the paper we use $n = |V(G)|$ and $m = |E(G)|$. An n-qubit circuit over CNOT gates is a *graphic parity network* for a graph G if for every edge (u, v) of G, the term $\{u, v\}$ appears in the annotated circuit and the final state of the wires is the same as the original state. A term is *singleton* or *binary* if its cardinality is one or two, respectively. For our purpose, this definition suffices, and we refer to Appendix A for a definition requiring more quantum background.

We start with proving lower bounds announced in Theorems 1.1 and 1.2. These proofs have two implications. First, we define a novel graph class that contains all chordal graphs, and show that all the graphs in this class admit the smallest possible graphic parity networks. Second, we propose a randomized algorithm for synthesizing graphic parity networks for general graphs.

2.1 Lower bounds

A parity network C can also be read from right to left, which defines another quantum circuit, called the *inverse* of C. Consider a wire with ℓ operations in C, which defines $\ell + 1$ terms. It turns out that the same wire in the inverse of C has the same number of terms, and they appear in exactly the inversed order as in C. This observation is formalized as the following proposition, which does not use any special properties of graphs and holds for all parity networks. For the sake of completeness, we provide a proof in Appendix B.

Proposition 2.1 (Folklore)**.** *The inverse of a parity network for a set is a parity network for the same set. For each qubit wire, the wire in the inversed parity network has exactly the same terms, and they appear in the reversed order.*

The following bounds the number of non-binary terms generated by a graphic parity network.

Lemma 2.2. *In any graphic parity network for a connected graph* G*, there must be at least* $n - 1$ *operations whose outcomes are not binary.*

Proof. Let ℓ be the size of the graphic parity network. For $i = 0, 1, \ldots, \ell$, we define a hypergraph H_i whose vertex set is $V(G)$ and whose edge set consists of all terms generated by the first i operations, and let $c(H_i)$ denote the number of components of H_i. By definition,

$$n = c(H_0) \geqslant c(H_1) \geqslant \cdots \geqslant c(H_\ell) = 1,$$

where $c(H_0) = n$ because H_0 is edgeless and $c(H_\ell) = 1$ because G is connected by assumption. In particular, $c(H_i)$ is either $c(H_{i-1})$ or $c(H_{i-1}) - 1$, and the second case can only happen when the ith operation is applied to two terms that are disjoint. Such an operation is called a plus-operation, and by the discussion above, there are at least $n - 1$ plus-operations.

We take the first $n - 1$ plus-operations of the graphic parity network, and denote them as C_1, C_2, ... C_{n-1}. For $i = 1, \ldots, n - 1$, we select a distinct operation of which the term on the target wire *before* the operation is non-binary. We take C_i if it satisfies our condition; otherwise, we take the next non-plus operation C'_i with the same target wire as C_i. Note that it exist because the final state of this wire is singleton. Since the term on the target wire is binary before C_i, and all operations between C_i and C'_i are plus-operations, the term on the target wire before C'_i is non-binary. All the $n - 1$ selected operations are distinct by the selection. In the inverse of this graphic parity network, these $n - 1$ non-binary terms appears *after* the operations, namely, there are $n - 1$ operations generating non-binary terms. And by theorem 2.1, if a term is generated in the inverse, then it must be generated in the origin, and this concludes the proof. □

Since there are at least m operations generating binary terms, Theorem 1.1 follows from Lemma 2.2 as a corollary. We say that a graphic parity network is *perfect* if its size is precisely $m + n - c$. By Lemma 2.2, there is a one-to-one mapping between $E(G)$ and the binary terms of a perfect graphic parity network. Moreover, if $m \gg n$, most of the terms for edges in G are generated by cancellation. Indeed, all chordal graphs admit perfect graphic parity networks. This can be extended to graphs close to chordal, e.g., the graph in Figure 2b, where $p = n - 2$. This graph can be turned into a chordal graph by adding a single edge, namely, $(1, 2)$, and hence it admits a graphic parity network of size $m + n = 3p + 3$. Note that all the induced cycles in $K_{2,p}$ have length four. If the girth of a graph is greater than four, the bound in Theorem 1.1 can be greatly improved.

Lemma 2.3. *Let* G *be a graph of girth at least five. The minimum size of graphic parity networks for* G *is* $m + \Omega(m)$.

Proof. Let us fix a graphic parity network for G. For each edge $e \in E(G)$, let $f(e)$ denote the operation that generates the term corresponding to e, and $c(e)$ the immediately next operation targeting the same wire as $f(e)$; note that $c(e)$ exists because the final state of this wire is a singleton term. Let $F = \{f(e) \mid e \in E(G)\}$ and $C = \{c(e) \mid e \in E(G)\}$. Moreover, let R denote all the operations not in F. Note that $|F| = |C| = m$, and the size of the network is

$$|F| + |R| \geqslant |F| + |C \setminus F| = |F| + |C| - |F \cap C| = 2m - |F \cap C|.$$

We are done if $|F \cap C| \leqslant m/2$. Hence, we assume that $|F \cap C| > m/2$. By definition, each operation in $F \cap C$ is $c(e_1) = f(e_2)$ for two different edges e_1 and e_2. There are two cases,

$$\begin{cases} \{v, w\} \dashrightarrow \oplus \{u, v\} & e_1 = (u, v), e_2 = (u, w), \\ \{u, v, w, x\} \dashrightarrow \oplus \{u, v\} & e_1 = (u, v), e_2 = (w, x), \end{cases}$$

where the three or four vertices involved in the operation are all distinct. We use F_1 and F_2 to denote the sets of these two types of operations. Note that $F \cap C = F_1 \cup F_2$.

Case 1. Since G does not contain any triangles, (v, w) is not an edge. In other words, (v, w) is generated by an operation in R. Moreover, if there is another operation $c(e_1') = f(e_2')$ in F_1 using (v, w), then $e_1' = (x, v)$ and $e_2' = (x, w)$ for some vertex $x \neq u$. But then $uvxw$ is a cycle of length four, violating the assumption. Therefore, each operation in F_1 corresponds to a distinct operation in R.

Case 2. The term $\{u, v, w, x\}$ is generated by an operation in R. There are at most three operations in F_2 using the term $\{u, v, w, x\}$. Therefore, there are at least $|F_2|/3$ such 4-terms generated by R.

In summary,

$$|R| \geqslant |F_1| + \frac{|F_2|}{3} \geqslant \frac{|F_1| + |F_2|}{3} = \frac{|F \cap C|}{3} > \frac{m}{6}.$$

This concludes the proof. □

Theorem 1.2 follows from Lemma 2.3 and the following lemma. The proof of Theorem 2.4, which is based on a classic result from extremal combinatorics, is deferred to the appendix.

Lemma 2.4. *For any positive integer n, there exists a graph G that has $\Omega(n\sqrt{n})$ edges and whose girth is at least five.*

A graphic parity network of size $m + n - c$ is called *perfect*, and a graph is called *perpane* if it admits a PERfect graphic PArity NEtwork.

2.2 Perfect cancellation graphs

Let $N(v)$ denote the neighborhood of v, and $N(U) = \bigcup_{v \in U} N(v) \setminus U$ for a vertex set $U \subseteq V(G)$. Let $\sigma : V(G) \mapsto [n]$ be an ordering of the vertices of G, where $[n] = \{1, 2, \ldots, n\}$. A subset $U \subseteq V(G)$ is σ-*linked* if every two consecutive vertices in $\sigma|_U$, the sub-ordering of σ induced by U, are adjacent in G. We use $x <_\sigma y$ (resp., $x \leqslant_\sigma y$) to denote $\sigma(x) < \sigma(y)$ (resp., $\sigma(x) \leqslant \sigma(y)$). For each vertex $v \in V(G)$, we denote

$$N_\sigma^+(v) = \{u \in N(v) \mid \sigma(u) > \sigma(v)\}.$$

Definition. *An ordering $\sigma : V(G) \mapsto [n]$ is a* perfect cancellation ordering *of G if for all vertices $v \in V(G)$ and for all components C of $G - v$, the set $N_\sigma^+(v) \cap C$ is σ-linked. A graph G is a* perfect cancellation graph *if it has a perfect cancellation ordering.*

The vertex set of a chordal graph can be ordered such that, each vertex v and its neighbors that occur after v in the order form a clique; such an order is called a *perfect elimination ordering* [29]. Since a clique is σ-linked for any ordering σ, a perfect elimination ordering is a perfect cancellation ordering [29]. In other words, all chordal graphs are perfect cancellation graphs. It is worth noting that a perfect cancellation ordering of a chordal graph is not necessarily a perfect elimination ordering. For example, both $(1,2,3,4)$ and $(4,3,2,1)$ are perfect cancellation orderings, but only the second is a perfect elimination ordering. With a perfect cancellation ordering of G given, we can synthesize a perfect graphic parity network for G in linear time. Indeed, the circuit in Figure 1b was generated by Algorithm 1. For the convenience of presentation, we may start with biconnected graphs, for which the condition is simplified to $N_\sigma^+(v)$ being σ-linked for all v.

Algorithm 1: A synthesizing algorithm for perfect cancellation graphs

```
1  for i ← 2, 3, ..., n do                    ▷ [v_1, ..., v_n] is a perfect cancellation ordering of G
2  |  for j ← i − 1, i − 2, ..., 1 do
3  |  |  if (v_i, v_j) ∈ E(G) then             ▷ j ∈ term(j) and |term(j)| ⩽ 2
4  |  |  |  if term(j) = {j} then
5  |  |  |  |  CNOT(i, j);
6  |  |  |  else
7  |  |  |  |  {j, k} ← term(j);               ▷ k > j and term(k) = {i, k}
8  |  |  |  |  CNOT(k, j);
9  for i ← n − 1, n − 2, ..., 1 do
10 |  {i, j} ← term(i);                        ▷ j > i and term(j) = {j}
11 |  CNOT(j, i);
```

Lemma 2.5. *Let G be a biconnected graph. Given a perfect cancellation ordering of a graph G, we can synthesize a perfect graphic parity network for G in $O(m+n)$ time.*

Proof. Let σ be the perfect cancellation ordering. We may number the vertices such that $\sigma(v_i) = i$, and use Algorithm 1. It generates all the binary terms in the main loop (lines 1–8) before restoring the singleton terms in line 9. Initially, $\mathrm{term}(j) = \{j\}$ for all $j = 1, \ldots, n$. The algorithm maintains the following invariants. Before the execution of the ith iteration, for all $j = 1, \ldots, n$,

(I1) $j \in \mathrm{term}(j)$ and $|\mathrm{term}(j)| \leqslant 2$;

(I2) $\mathrm{term}(j) = \{j\}$ if $j \geqslant i$; and

(I3) if $\mathrm{term}(j) = \{j, k\}$, then $j < k < i$, $(v_j, v_k) \in E(G)$, and $(v_j, v_{k'}) \notin E(G)$ for all k' with $k < k' < i$.

Now we show that the invariants are maintained. In the iteration from the inner loop (lines 2 to 8), only $\mathrm{term}(j)$ is modified. By invariant (I2), $\mathrm{term}(i) = \{i\}$, and it remains true during the ith iteration of the for the main loop because wire i is never the target. Moreover, for each $j < i$, wire j is the target in and only in the jth iteration of the inner loop. If $(v_i, v_j) \notin E(G)$, then $\mathrm{term}(j)$ is not changed, and all the invariants remain true. Hence, assume $(v_i, v_j) \in E(G)$, and we argue that $\mathrm{term}(j) = \{i, j\}$ after this iteration, and then all invariants remain satisfied afterward. It is straightforward when $\mathrm{term}(j) = \{j\}$ (line 5), and we focus on the else branch (line 6). Line 7 is correct by (I1) and the fact that the condition in line 4 is not satisfied. By invariant (I3), $k > j$, and $(v_{k'}, v_j) \notin E(G)$ for all k' with $k < k' < i$. Since σ is a perfect

cancellation ordering, k and i are adjacent. By invariant (I3), $\mathrm{term}(k) = \{k, i\}$, and thus line 8 sets $\mathrm{term}(j)$ to $\{k, i\}$. Thus, the algorithm correctly produces a graphic parity network for G.

We now verify that the synthesized circuit is perfect. For each edge, the algorithm introduces precisely one gate. Moreover, since G is connected, by invariant (I3), $|\mathrm{term}(j)| = 2$ for all $j = 1, \ldots, n-1$ when the algorithm reaches line 9. By invariants (I1) and (I3), we can restore every wire to be a singleton term by one gate. Thus, the total size is precisely $m + n - 1$.

Let us briefly explain the implementation. We assume the graph is stored as adjacency lists. It is pedestrian to reconstruct the lists such that each list is sorted. Thus, the number of iterations of the loop of line 3 can be the number of neighbors of v. The total time is thus $O(m + n)$ □

If G is not biconnected, we synthesize a graphic parity network for G by independently handling its biconnected components. The detailed description as well as the formal proof of theorem 1.4 is deferred to appendix D.

2.3 A randomized synthesizing algorithm

Let G be an arbitrary graph. We present a random algorithm to synthesize a graphic parity network for G. We process the vertices in a random order, and for each vertex i, we generate all the edges between this vertex and latter vertices in this order, before resetting this wire to $\{i\}$. Similar to Algorithm 1, we never introduce a term with more than two elements, and the item i never leaves wire i. It has three phases.

The first phase only applies when the term on wire i is binary, i.e., it has an earlier neighbor in the order. Let j be the other number in this term. For all unprocessed neighbors v_k of v_i, if the term on wire k is $\{j, k\}$, we can generate $\{i, k\}$ by adding $\{i, j\}$ to it. The three vertices form a triangle.

In the second phase, we deal with neighbors v_j of v_i such that the term on wire j is binary. We group them according to the other item in their terms, and process each group by cancellation. Here we attempt to add a non-edge term $\{i, j\}$ to facilitate dealing with edges between v_i and common neighbors of v_i and v_j; see the discussion about $K_{2,p}$ above for motivation.

Finally, we deal with other neighbors of v_i individually. We summarize it as Algorithm 2.

Lemma 2.6. *Algorithm 2 synthesizes a graphic parity network for* G *in polynomial time.*

Proof. The algorithm starts with renumbering the vertices such that $\pi(v_i) = i$. Initially, $\mathrm{term}(i) = \{i\}$ for all $i = 1, \ldots, n$. In the ith iteration of the main loop (lines 2–19), the algorithm generates all the terms for edges (v_i, v_k) with $k > i$, before reseting $\mathrm{term}(i) = \{i\}$. If $\mathrm{term}(i)$ is binary at the beginning, line 8 resets it. The only operations targeting wire i are lines 12 and 16. If line 12 changes wire i, line 16 duly resets it. It remains to verify that all edges are generated. Let v_k be a neighbor of v_i with $k > i$. If $\mathrm{term}(k)$ is not binary before the ith main loop, it is added by either line 6 or line 15, depending upon whether $\mathrm{term}(i)$ and $\mathrm{term}(k)$ have a item. Thus, the algorithm correctly produces a graphic parity network for G. The algorithm clearly runs in polynomial time. □

In Algorithm 2, except for lines 12 and 16, each other operation either generates a new item or clears up a wire. Therefore, to bound the size of the synthesized circuit, it suffices to bound the number of them being executed. Since they are always executed in pair, it suffices to count line 12. For a set X, we use $\mathfrak{S}_X$ to denote the set of all permutations of X, and we use $\mathfrak{S}_n$ as an shorthand for $\mathfrak{S}_{[n]}$. The degree of vertex v is $d(v)$.

Algorithm 2: A randomized algorithm for graphic parity network synthesis.

1 renumber the vertices such that $\pi(v_i) = i$; ▷ *π is a random permutation of* $[n]$.
2 **for** $i \leftarrow 1, 2, \ldots, n$ **do** ▷ *The size of each term is at most two.*
3 **if** term(i) *is binary* **then**
4 $\{i, j\} \leftarrow$ term(i); ▷ $j < i$.
5 **for** $v_k \in N(v_i)$ ***such that*** term$(k) = \{j, k\}$ **do** ▷ $k > i$.
6 CNOT(i, k);
7 remove edge (v_i, v_k);
8 CNOT(j, i);
9 **for** $j \leftarrow 1, 2, \ldots, i-1$ **do**
10 $K \leftarrow \{v_k \in N(v_i) \mid \text{term}(k) = \{j, k\}\}$;
11 **if** $K \neq \emptyset$ **then** ▷ $v_i v_j \notin E(G)$ *or is already generated.*
12 CNOT(j, i);
13 **for** $v_k \in K$ **do** ▷ $k > i$.
14 CNOT(i, k);
15 remove edge (v_i, v_k);
16 CNOT(j, i) ▷ *Clean up.*
17 **for** $v_k \in N(v_i)$ **do** ▷ $k > i$.
18 CNOT(i, k);
19 remove edge (v_i, v_k);

Lemma 2.7. *The expected number of line 12 being executed is upper bounded by*

$$4n + \min_{1 \leqslant t \leqslant n} \left\{ 2 \sum_{i<t} d_i + \frac{(n-t)n \log n}{d_t} \right\},$$

where $d_1, d_2, \ldots, d_n$ *are the degrees of the vertices sorted in ascending order.*

Proof. Line 12 is only executed when $K \neq \emptyset$ and $i \notin$ term(j) at the beginning of the ith iteration. With permutation π, the number of line 12 being executed is the number of pairs (u, v) such that there exists an extra vertex w satisfying

(i) $uw, vw \in E(G)$;

(ii) u is not the last neighbor of v in π;

(iii) $\pi(u) < \pi(v) < \pi(w)$; and

(iv) $wx \notin E(G)$ for all vertices x with $\pi(u) < \pi(x) < \pi(v)$.

Therefore, it cannot be more than the number of pairs (u, v) such that there exists an extra vertex w merely satisfying (iv) and

(iv) $uw, vw \in E(G)$ and $\pi(u) < \pi(v)$.

Now we obtain an upper bound on the number of such pairs.

For any permutation $\pi \in \mathfrak{S}_n$, let $P(\pi)$ denote all such pairs when the algorithm is executed using permutation π, and let $P(\pi, j)$ denote the subset of $P(\pi)$ in which the first vertex is fixed

by $\pi(u) = j$. Then the expected number of such pairs is

$$\begin{aligned}
\mathop{\mathbb{E}}_{\pi\in\mathfrak{S}_n} |P(\pi)| &= \mathop{\mathbb{E}}_{\pi\in\mathfrak{S}_n} \sum_{j=1}^{n} |P(\pi, j)| \\
&= \sum_{j=1}^{n} \mathop{\mathbb{E}}_{\pi\in\mathfrak{S}_n} |P(\pi, j)| \\
&= \sum_{j=1}^{n} \mathop{\mathbb{E}}_{X\in\binom{[n]}{j}} \mathop{\mathbb{E}}_{\substack{\pi\in\mathfrak{S}_X \\ \forall x\in X.\pi(x)>n-j}} |P(\pi, n-j+1)| \\
&\leqslant \sum_{j=1}^{n} \mathop{\mathbb{E}}_{\pi\in\mathfrak{S}_n} |P(\pi, 1)|. \qquad (1)
\end{aligned}$$

Therefore, it reduces to bounding the expected size of $P(\pi, 1)$. Consider any fixed $t \in [n]$. If $d(\pi^{-1}(1)) \leqslant d_t$, then

$$|P(\pi, 1)| \leqslant d_t. \qquad (2)$$

We now consider the nontrivial case, where $d(\pi^{-1}(1)) > d_t$. Note that $|P(\pi, 1)|$ is the number of vertices v such that there exists another vertex w whose first two neighbors in π are $\pi^{-1}(1)$ and v; we say that it is *witnessed by* w. For each vertex $w \in N(\pi^{-1}(1))$, let X_w be the index of the second neighbor of x in π. Then

$$\mathbb{P}(X_w = i) = \left(1 - \frac{d(w)-1}{n-2}\right) \cdots \left(1 - \frac{d(w)-1}{n-(i-1)}\right) \frac{d(w)-1}{n-i} > \left(1 - \frac{d(w)-1}{n}\right)^{i-2} \frac{d(w)-1}{n}.$$

Letting $\alpha = 1 - \frac{d(w)-1}{n}$, we have

$$\begin{aligned}
&\mathbb{P}\left(X_w \leqslant \frac{n\log n}{d(w)-1}\right) \\
=&\mathbb{P}(X_w = 2) + \cdots + \mathbb{P}\left(X_w = \left\lfloor \frac{n\log n}{d(w)-1} \right\rfloor\right) \\
>&\frac{d(w)-1}{n} + \cdots + \alpha^{\left\lfloor \frac{n\log n}{d(w)-1} \right\rfloor - 2} \left(\frac{d(w)-1}{n}\right) \\
=&\left(\frac{1-\alpha^{\left\lfloor \frac{n\log n}{d(w)-1} \right\rfloor - 1}}{1-\alpha}\right) \left(\frac{d(w)-1}{n}\right) \\
=&1 - \alpha^{\left\lfloor \frac{n\log n}{d(w)-1} \right\rfloor - 1}.
\end{aligned}$$

If $d(w) > d_t$, then $\frac{n\log n}{d(w)-1} \leqslant \frac{n\log n}{d_t}$, and

$$\mathbb{P}\left(X_w > \frac{n\log n}{d_t}\right) < \alpha^{\left\lfloor \frac{n\log n}{d(w)-1} \right\rfloor - 1} = \left(1 - \frac{d(w)-1}{n}\right)^{\left\lfloor \frac{n\log n}{d(w)-1} \right\rfloor - 1} < \frac{4}{n}. \qquad (3)$$

Since the number of such vertices w is less than $n-1$, then all such vertices witness at most $\frac{n\log n}{d_t} + 4$ vertices. On the other hand, each vertex w with $d(w) \leqslant d_t$ witnesses at most one vertex. In summary, for each $u \in V(G)$,

$$\mathop{\mathbb{E}}_{\substack{\pi\in\mathfrak{S}_n \\ \pi(u)=1}} |P(\pi, 1)| < \frac{n\log n}{d_t} + 4 + \sum_{\substack{w\in N(u) \\ d(w)\leqslant d_t}} 1. \qquad (4)$$

Combining (1), (2), and (4), we conclude that the expected number of line 12 being executed is less than

$$\sum_{d_i \leqslant t} d_i + \sum_{\substack{u \in V(G) \\ d(u) > d_t}} \left(\frac{n \log n}{d_t} + 4 + \sum_{\substack{w \in N(u) \\ d(w) \leqslant d_t}} 1 \right) = 2 \sum_{d_i \leqslant t} d_i + \sum_{d_i > t} \frac{n \log n}{d_t} + 4n. \tag{5}$$

The statement follows because (5) holds for all $t \in [n]$. □

Theorem 1.3 is thus a direct consequence of Theorem 2.6 and Theorem 2.7, whose proof is defered to appendix E.

3 Recognition of perfect cancellation graphs

For any graph class, the first algorithmic question is its *recognition*: to decide whether a given graph is in this class. In this section, we investigate the complexity and algorithms of recognizing perfect cancellation graphs.

3.1 The NP-completeness of recognition

We now prove theorem 1.5, showing that the recognition of perfect cancellation graphs is NP-compete, by a reduction from the following problem, which is shown to be NP-complete by Opatrny [26].

Definition (Betweeness)**.** *Given a finite set S and a set of ordered triples $T \subseteq S \times S \times S$, the betweenness problem asks to determine whether there exists a total ordering π of S such that for every triple (x, y, z) in T, either $x <_\pi y <_\pi z$ or $z <_\pi y <_\pi x$.*

The key observation of our reduction is to use a set of false twins to force the order on a triple of vertices with two edges among them. A set of vertices with the same neighborhood is called a *false twins*. Note that by definition, there cannot be any edge among false twins.

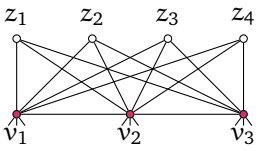
Lemma 3.1. *Let G be a perfect cancellation graph and I a set of false twins of G. If $|N(I)| \leqslant |I| - 1$, then $N(I)$ is σ-linked in any perfect cancellation ordering σ of G,*

Proof. The statement holds vacuously if I comprises a single vertex. Hence, we assume that $|I| \geqslant 2$, and hence no vertex in I is a cut vertex. Let v be the first vertex in σ from $I \cup N(I)$. It suffices to show that $v \in I$: note that $N^+_\sigma(v) = N(v) = N(I)$. Suppose for contradiction that $v \notin I$, i.e., $v \in N(I)$. We may number the vertices in $N^+_\sigma(v)$ as $u_1, \ldots, u_\ell$, where $\ell = |N^+_\sigma(v)|$, such that $u_i <_\sigma u_{i+1}$ for all $i = 1, \ldots, \ell - 1$. Since there is no edge among vertices in I, between any two of them there is another vertex, which has to be from $N(I)$. This is nevertheless impossible because there are at most $|N(I) \setminus \{v\}| \leqslant |I| - 2$ such vertices. □

In Figure 3, for example, in any perfect cancellation ordering σ, either $v_1 <_\sigma v_2 <_\sigma v_3$ or $v_3 <_\sigma v_2 <_\sigma v_1$.

We are now ready to describe the reduction from an instance (S, T) of the betweenness problem to the recognition of perfect cancellation graphs. Let $p = |S|$ and $q = |T|$. We construct a graph G on $2p + 14q$ vertices as follows. First, for each element x in S, introduce a vertex; abusing notation, we use x to denote both the element and the corresponding vertex, and use S to denote this set of vertices. Second, we introduce a vertex set

$$C = \{v_1^i, v_3^i \mid 1 \leqslant i \leqslant q\}.$$



.11 w

Figure 3: Illustration for Theorem 3.1, where $I = \{z_1, \ldots, z_4\}$ and $v_1v_2v_3$ is an induced path. Note that v_1, v_2, v_3 might have other neighbors.

We add an edge between each pair of vertices in C unless they have the same superscript (note that the complement of the subgraph induced by C is an induced matching), and we add all the $2pq$ edges between S and C. Third, we add a set U of p vertices, and make them universal in the subgraph induced by $S \cup C \cup U$.

Finally, we add the sets of false twins to enforce the desired order. For convenience, we use (v_0^i, v_2^i, v_4^i) to denote the three vertices in S corresponding to the three elements in the ith triple in T in order. For each $i = 1, \ldots, q$, we introduce 12 vertices $z_1^i, \ldots, z_{12}^i$. For each $j = 0, 1, 2$, let

$$Z_j^i = \{z_{4j+1}^i, \ldots, z_{4j+4}^i\},$$

and add all the 12 edges between Z_j^i and $\{v_j^i, v_{j+1}^i, v_{j+2}^i\}$. See Figure 4 for an illustration.

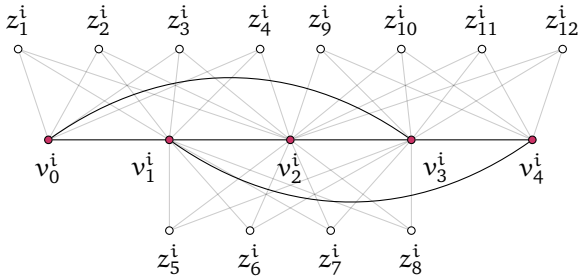


Figure 4: The construction for the proof of Theorem 1.5.

Proof of Theorem 1.5. It is easy to check whether an ordering is a perfect cancellation ordering, the recognition of perfect cancellation graphs is in **NP**. For its **NP**-hardness, we show that (S, T) is a yes-instance of the betweenness problem if and only if the graph G constructed above is a perfect cancellation graph.

For sufficiency, suppose that G is a perfect cancellation graph, and let $\sigma : V(G) \mapsto [2p + 14q]$ be a perfect cancellation ordering of G. For $i = 1, \ldots, q$, Theorem 3.1 applied to the set Z_1^i forces that either $v_1^i <_\sigma v_2^i <_\sigma v_3^i$ or $v_3^i <_\sigma v_2^i <_\sigma v_1^i$. We may assume without loss of generality that

$$v_1^i <_\sigma v_2^i <_\sigma v_3^i,$$

and the other is symmetric. Then Theorem 3.1 applied to sets Z_0^i and Z_2^i forces that

$$v_0^i <_\sigma v_1^i <_\sigma v_2^i \text{ and } v_2^i <_\sigma v_3^i <_\sigma v_4^i,$$

Thus,

$$v_0^i <_\sigma v_2^i <_\sigma v_4^i,$$

and $\pi = \sigma|_S$ is a valid ordering of S that certificates that (S, T) is a yes-instance.

For necessity, suppose that that (S, T) is an yes-instance of the betweenness problem, and let π be a valid ordering of S. We may number the elements in S such that $\pi = \langle x_1, \ldots, x_p \rangle$. Since reversing triples in T does not change the instance, we may assume without loss of generality that for all $i = 1, \ldots, q$,

$$v_0^i <_\pi v_2^i <_\pi v_4^i.$$

For $i = 1, \ldots, p$, let $c(i)$ denote the number such that $x_{c(i)} = v_2^i$; note that

$$1 < c(i) < p.$$

We number the vertices in U as $\{u_1, \ldots, u_p\}$.

We construct a perfect cancellation ordering σ of $V(G)$ as follows. It starts from

$$z_1^1, \ldots, z_{12}^1, \ldots, z_{12}^q, u_1, x_1, \ldots, u_p, x_p.$$

For all $i = 1, \ldots, q$, we put v_1^i in between $x_{c(i)-1}$ and $u_{c(i)}$ and v_3^i in between $x_{c(i)}$ and $u_{c(i)+1}$; vertices assigned to the same range are in an arbitrary order. Note that

$$\sigma(x_i) = 12q + 2i + 2|\{j \mid c(j) < i\}| + |\{j \mid c(j) = i\}|.$$

Now we verify that σ is a perfect cancellation ordering of G. By construction,

$$v_0^i \leqslant_\sigma x_{c(i)-1} <_\sigma v_1^i <_\sigma x_{c(i)} = v_2^i <_\sigma v_3^i <_\sigma x_{c(i)+1} \leqslant_\sigma v_4^i.$$

Thus, $N_\sigma^+(z_j^i) = N(z_j^i)$ is σ-linked for all $i = 1, \ldots, q$ and $j = 1, \ldots, 12$. If two vertices after z_{12}^q in σ are not adjacent, they are either x_j and x_i for $1 \leqslant j < i \leqslant q$, or v_1^i and v_3^i for some i. Such a pair is always separated by the vertex u_i. Since u_i is universal in the subgraph induced by $S \cup C \cup U$, the ordering σ is a perfect cancellation ordering of G. □

3.2 Recognition is fixed-parameter tractable

Without loss of generality, we assume that the input graph is biconnected. Let k be the treewidth of the input graph, and let us assume we have a *nice* tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of width k (definition in Appendix F). During the bottom-up dynamic programming process, we can partition the vertex into P, C, and F, which denotes the set of forgotten vertices (past), the set of vertices in the current bag (current), and the set of vertices that we have not met (future). By definition, there is no edge between P and F. If σ is a perfect cancellation ordering, then for any vertex v, we have a path on $\sigma|_{N_\sigma^+(v)}$. It is broken into several sub-paths with vertices in P removed, whose ends (except the original ends of the path) are all from C.

Definition (Valid order)**.** *Let t be a node of T. A permutation σ of V_t is valid for t if*

(V1) for each $v \in V_t \setminus X_t$, the set $N_\sigma^+(v)$ is σ-linked; and

(V2) for each $v \in X_t$, if two consecutive vertices in $\sigma|_{N_\sigma^+(v)}$ are not adjacent, they must be in X_t.

We maintain the set of all valid orders for each node t in T, using the *canonical representation* which allows us to keep the solution space small (namely, with size of a function of the treewidth). More details can be found in the Appendix F.

4 Concluding remarks

The main open problem is whether there are perpane graphs that are not perfect cancellation graphs.

Conjecture 1. *A graph is perpane if and only if it is a perfect cancellation graph.*

We know that there are perfect graphic parity networks that do not correspond to any ordering in an apparent way. Thus, to answer the conjecture, we need to better understand perfect graphic parity networks, for which there are several open questions. In particular, whether a perpane graph always admits

(C1) a perfect graphic parity network in which all terms are singleton and binary;

(C2) a perfect graphic parity network in which some wire is not a control of any operation; and

(C3) a perfect graphic parity network in which no qubit leaves its original wire.

For all the properties, we have examples of perfect graphic parity networks violating them. But we can find alternative perfect graphic parity networks with the desired properties. Note that Theorem 2.2 does not rule out the existence of terms of cardinality three or more, though it does imply that if there is such a term, there are more wires that remain singleton throughout.

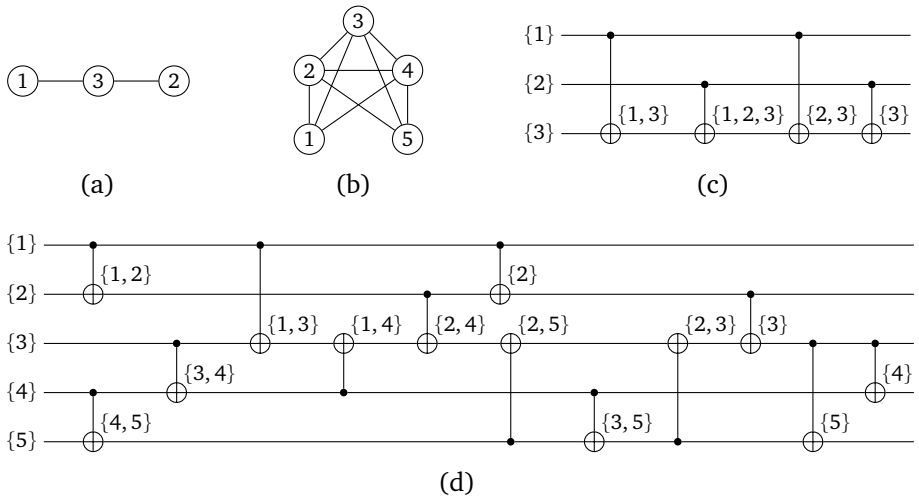


Figure 5: Perfect graphic parity networks that do not satisfy the conditions. (a) A line of length 3. (b) A chordal graph. (c) A perfect graphic parity network for (a) violating (C1). (d) A perfect graphic parity network for (b) violating (C2), (C3).

If a graph contains a set U of $\lfloor \frac{n}{2} \rfloor$ universal vertices, then we can make a perfect cancellation ordering by arranging the vertices in $V(G) \setminus U$ and in U alternatively.

Proposition 4.1. *A graph containing $\lfloor \frac{n}{2} \rfloor$ universal vertices is a perfect cancellation graph.*

An interesting question arising from Theorem 4.1 is: given a graph, what is the minimum number of universal vertices we need to add to make it a perfect cancellation graph? Even more interesting is adding edges. Since all chordal graphs are perfect cancellation graphs, this cannot be larger than the well-studied chordal completion number [6, 18]. Can we always turn a graph into a perfect cancellation graph by adding at most m edges? We would not bother if it needs more than m: any graph has a trivial graphic parity network of size $2m$.

References

[1] Matthew Amy, Parsiad Azimzadeh, and Michele Mosca. On the controlled-NOT complexity of controlled-NOT–phase circuits. *Quantum Science and Technology*, 4(1):015002, September 2018. doi:10.1088/2058-9565/aad8ca.

[2] Anurag Anshu and Tony Metger. Concentration Bounds for Quantum States and Limitations on the QAOA from Polynomial Approximations. In *DROPS-IDN/v2/Document/10.4230/LIPIcs.ITCS.2023.5*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.5.

[3] Joao Basso, Edward Farhi, Kunal Marwaha, Benjamin Villalonga, and Leo Zhou. The Quantum Approximate Optimization Algorithm at High Depth for MaxCut on Large-Girth Regular Graphs and the Sherrington-Kirkpatrick Model. In *DROPS-IDN/v2/Document/10.4230/LIPIcs.TQC.2022.7*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.TQC.2022.7.

[4] Joao Basso, David Gamarnik, Song Mei, and Leo Zhou. Performance and limitations of the QAOA at constant levels on large sparse hypergraphs and spin glass models. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 335–343, October 2022. doi:10.1109/FOCS54457.2022.00039.

[5] Sergey Bravyi, Andrew W. Cross, Jay M. Gambetta, Dmitri Maslov, Patrick Rall, and Theodore J. Yoder. High-threshold and low-overhead fault-tolerant quantum memory. *Nature*, 627(8005):778–782, March 2024. doi:10.1038/s41586-024-07107-7.

[6] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.

[7] Sirui Cao, Bujiao Wu, Fusheng Chen, Ming Gong, Yulin Wu, Yangsen Ye, Chen Zha, Haoran Qian, Chong Ying, Shaojun Guo, Qingling Zhu, He-Liang Huang, Youwei Zhao, Shaowei Li, Shiyu Wang, Jiale Yu, Daojin Fan, Dachao Wu, Hong Su, Hui Deng, Hao Rong, Yuan Li, Kaili Zhang, Tung-Hsun Chung, Futian Liang, Jin Lin, Yu Xu, Lihua Sun, Cheng Guo, Na Li, Yong-Heng Huo, Cheng-Zhi Peng, Chao-Yang Lu, Xiao Yuan, Xiaobo Zhu, and Jian-Wei Pan. Generation of genuine entanglement up to 51 superconducting qubits. *Nature*, 619(7971):738–742, July 2023. doi:10.1038/s41586-023-06195-1.

[8] Pafnuty Lvovich Chebyshev. Mémoire sur les nombres premiers. *Journal de mathématiques pures et appliquées*, Série 1:366–390, 1852. Proof of the postulate: 371–382.

[9] Timothée Goubault de Brugière and Simon Martiel. Faster and shorter synthesis of Hamiltonian simulation circuits, April 2024. arXiv:2404.03280.

[10] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. *arXiv:1411.4028 [quant-ph]*, November 2014. arXiv:1411.4028.

[11] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum Computation by Adiabatic Evolution, January 2000. arXiv:quant-ph/0001106.

[12] Vlad Gheorghiu, Jiaxin Huang, Sarah Meng Li, Michele Mosca, and Priyanka Mukhopadhyay. Reducing the CNOT Count for Clifford+T Circuits on NISQ Architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(6):1873–1884, June 2023. doi:10.1109/TCAD.2022.3213210.

[13] Timothée Goubault de Brugière and Simon Martiel. Shallower cnot circuits on realistic quantum hardware. *ACM Transactions on Quantum Computing*, November 2024. Just Accepted. doi:10.1145/3700884.

[14] Matthew P. Harrigan, Kevin J. Sung, Matthew Neeley, Kevin J. Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C. Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B. Buckley, David A. Buell, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Sean Demura, Andrew Dunsworth, Daniel Eppens, Austin Fowler, Brooks Foxen, Craig Gidney, Marissa Giustina, Rob Graff, Steve Habegger, Alan Ho, Sabrina Hong, Trent Huang, L. B. Ioffe, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Cody Jones, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Seon Kim, Paul V. Klimov, Alexander N. Korotkov, Fedor Kostritsa, David Landhuis, Pavel Laptev, Mike Lindmark, Martin Leib, Orion Martin, John M. Martinis, Jarrod R. McClean, Matt McEwen, Anthony Megrant, Xiao Mi, Masoud Mohseni, Wojciech Mruczkiewicz, Josh Mutus, Ofer Naaman, Charles Neill, Florian Neukart, Murphy Yuezhen Niu, Thomas E. O'Brien, Bryan O'Gorman, Eric Ostby, Andre Petukhov, Harald Putterman, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Andrea Skolik, Vadim Smelyanskiy, Doug Strain, Michael Streif, Marco Szalay, Amit Vainsencher, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Leo Zhou, Hartmut Neven, Dave Bacon, Erik Lucero, Edward Farhi, and Ryan Babbush. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nature Physics*, 17(3):332–336, March 2021. doi:10.1038/s41567-020-01105-y.

[15] Jiaqing Jiang, Xiaoming Sun, Shang-Hua Teng, Bujiao Wu, Kewen Wu, and Jialin Zhang. Optimal space-depth trade-off of cnot circuits in quantum logic synthesis. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, pages 213–229, USA, 2020. Society for Industrial and Applied Mathematics.

[16] Hopcroft John and Tarjan Robert. Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM*, June 1973. doi:10.1145/362248.362272.

[17] John Kallaugher, Ojas Parekh, and Nadezhda Voronova. Exponential Quantum Space Advantage for Approximating Maximum Directed Cut in the Streaming Model. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1805–1815, Vancouver BC Canada, June 2024. ACM. doi:10.1145/3618260.3649709.

[18] Haim Kaplan, Ron Shamir, and Robert Endre Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM Journal on Computing*, 28(5):1906–1922, 1999. A preliminary version appeared in FOCS 1994. doi:10.1137/S0097539796303044.

[19] Ferenc Kárteszi. *Introduction to Finite Geometries*. North-Holland Publishing Company, 1976.

[20] Zhiyuan Li, Pei Liu, Peng Zhao, Zhenyu Mi, Huikai Xu, Xuehui Liang, Tang Su, Weijie Sun, Guangming Xue, Jing-Ning Zhang, Weiyang Liu, Yirong Jin, and Haifeng Yu. Error per single-qubit gate below 10^{-4} in a superconducting qubit. *npj Quantum Information*, 9(1):111, 2023. doi:10.1038/s41534-023-00781-x.

[21] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2, 2014. URL: http://journal.frontiersin.org/article/10.3389/fphy.2014.00005/abstract, doi:10.3389/fphy.2014.00005.

[22] Dmitri Maslov and Ben Zindorf. Depth optimization of cz, cnot, and clifford circuits. *IEEE Transactions on Quantum Engineering*, 3:1–8, 2022. doi:10.1109/TQE.2022.3180900.

[23] Mikko Möttönen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa. Quantum Circuits for General Multiqubit Gates. *Physical Review Letters*, 93(13):130502, sep 2004. URL: https://link.aps.org/doi/10.1103/PhysRevLett.93.130502, doi:10.1103/PhysRevLett.93.130502.

[24] Yunseong Nam, Neil J. Ross, Yuan Su, Andrew M. Childs, and Dmitri Maslov. Automated optimization of large quantum circuits with continuous parameters. *npj Quantum Information*, 4(1):1–12, May 2018. doi:10.1038/s41534-018-0072-4.

[25] Zhongchu Ni, Sai Li, Xiaowei Deng, Yanyan Cai, Libo Zhang, Weiting Wang, Zhen-Biao Yang, Haifeng Yu, Fei Yan, Song Liu, Chang-Ling Zou, Luyan Sun, Shi-Biao Zheng, Yuan Xu, and Dapeng Yu. Beating the break-even point with a discrete-variable-encoded logical qubit. *Nature*, 616(7955):56–60, April 2023. doi:10.1038/s41586-023-05784-4.

[26] Jaroslav Opatrny. Total Ordering Problem. *SIAM Journal on Computing*, 8(1):111–114, 1979. doi:10.1137/0208008.

[27] Guido Pagano, Aniruddha Bapat, Patrick Becker, Katherine S. Collins, Arinjoy De, Paul W. Hess, Harvey B. Kaplan, Antonis Kyprianidis, Wen Lin Tan, Christopher Baldwin, Lucas T. Brady, Abhinav Deshpande, Fangli Liu, Stephen Jordan, Alexey V. Gorshkov, and Christopher Monroe. Quantum approximate optimization of the long-range Ising model with a trapped-ion quantum simulator. *Proceedings of the National Academy of Sciences*, 117(41):25396–25401, October 2020. doi:10.1073/pnas.2006373117.

[28] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5, 2014. doi:10.1038/ncomms5213.

[29] Donald J Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(3):597–609, December 1970. doi:10.1016/0022-247X(70)90282-9.

[30] Ruslan Shaydulin, Changhao Li, Shouvanik Chakrabarti, Matthew DeCross, Dylan Herman, Niraj Kumar, Jeffrey Larson, Danylo Lykov, Pierre Minssen, Yue Sun, Yuri Alexeev, Joan M. Dreiling, John P. Gaebler, Thomas M. Gatterman, Justin A. Gerber, Kevin Gilmore, Dan Gresh, Nathan Hewitt, Chandler V. Horst, Shaohan Hu, Jacob Johansen, Mitchell Matheny, Tanner Mengle, Michael Mills, Steven A. Moses, Brian Neyenhuis, Peter Siegfried, Romina Yalovetzky, and Marco Pistoia. Evidence of scaling advantage for the quantum approximate optimization algorithm on a classically intractable problem. *Science Advances*, May 2024. doi:10.1126/sciadv.adm6761.

[31] Masuo Suzuki. General theory of fractal path integrals with applications to many-body theories and statistical physics. *Journal of Mathematical Physics*, 32(2):400–407, February 1991. doi:10.1063/1.529425.

[32] Juha J Vartiainen, Mikko Möttönen, and Martti M Salomaa. Efficient Decomposition of Quantum Gates. *Physical Review Letters*, 92(17):177902, 2004. URL: https://link.aps.org/doi/10.1103/PhysRevLett.92.177902, doi:10.1103/PhysRevLett.92.177902.

[33] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. Quantum approximate optimization algorithm for MaxCut: A fermionic view. *Physical Review A*, 97(2):022304, February 2018. doi:10.1103/PhysRevA.97.022304.

[34] Bujiao Wu, Xiaoyu He, Shuai Yang, Lifu Shou, Guojing Tian, Jialin Zhang, and Xiaoming Sun. Optimization of CNOT circuits on limited-connectivity architecture. *Physical Review Research*, 5(1):13065, 2023. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.5.013065, doi:10.1103/PhysRevResearch.5.013065.

[35] Jonathan Wurtz and Peter Love. MaxCut quantum approximate optimization algorithm performance guarantees for p $>$ 1. *Physical Review A*, 103(4):042612, April 2021. doi:10.1103/PhysRevA.103.042612.

[36] Jonathan Wurtz and Danylo Lykov. Fixed-angle conjectures for the quantum approximate optimization algorithm on regular MaxCut graphs. *Physical Review A*, 104(5):052419, November 2021. doi:10.1103/PhysRevA.104.052419.

[37] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices. *Physical Review X*, 10(2):021067, June 2020. doi:10.1103/PhysRevX.10.021067.

A More quantum background

Let $x, b \in \mathbb{F}_2^n$, then $x \cdot b = x_1 b_1 \oplus x_2 b_2 \oplus \cdots \oplus x_n b_n$.

Definition ([1]). *Given $S \subseteq \mathbb{F}_2^n$, a* parity network *for S is an n-qubit* CNOT *circuit that, with initial state $|b\rangle$,*

- *for all $x \in S$ the state $|x \cdot b\rangle$ appears on a certain qubit in the circuit; and*
- *the final state is $|b_1, b_2, \ldots, b_n\rangle$.*

The second requirement is from quantum algorithms like QAA and QAOA, where the original state must be restored at the end of the circuit. In applications like QAOA, the output is allowed to be a permutation of the input, i.e., the final state can be $\left|b_{\pi(1)}, b_{\pi(2)}, \ldots, b_{\pi(n)}\right\rangle$ for some permutation $\pi \in \mathfrak{S}_n$. But allowing this seems not only to fail to aid in resolving the problem but also to further complicate it, so our definition restricts the output to be exactly the same as the input.

We now briefly explain the use of graphic parity networks in optimizing the quantum circuits for QAA and QAOA. Recall the following formulation of the maximum cut problem:

$$\max_{x \in \{-1,1\}^{|V(G)|}} \sum_{(u,v) \in E(G)} \frac{1 - x_u x_v}{2}.$$

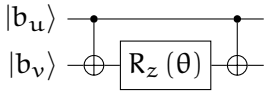
In the Ising formulation [21], it is equivalent to finding the ground energy (lowest eigenvalue) of the following Hamiltonian:

$$H_C = \sum_{(u,v) \in E} Z_u Z_v,$$

where Z_u stands for the Pauli Z operator acting on the qubit u. Both QAA as well as QAOA use the quantum adiabatic evolution to calculate the ground energy of H_C as follows.

1. Select a Hamiltonian H_B such that H_B has a simple ground state.
2. Initialize the quantum state to the ground state of H_B.
3. Evolve the quantum state under a time-dependent Hamiltonian that gradually changes from H_B to H_C.
4. The final state of the quantum system is the ground state of H_C, whose energy can be measured to obtain the solution to the maximum cut problem.

In the quantum circuit model, for the evolving of quantum state under a gradually changing Hamiltonian, one common approach is Trotter–Suzuki decomposition [31], where $\exp(iH_B\theta)$ and $\exp(iH_C\theta)$ should be implemented. For the implementation of $\exp(iH_C\theta)$, the parity networks come into play. For example, the naive implementation of $\exp(iH_C\theta)$ can be done by applying the following sequence of operations for all $(u, v) \in E(G)$:



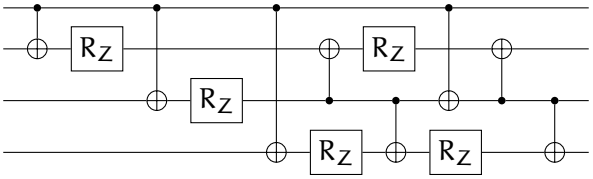


Figure 6: The corresponding circuit of the parity network in fig. 1b. It is generated by inserting a R_z gate whenever and wherever a new parity term is generated.

where R_z is the single qubit gate rotating along Pauli-Z axis. Suppose the input for the circuit is $|b_u, b_v\rangle$, then the output will be $\exp(-i(b_u \oplus b_v)\theta)|b_u, b_v\rangle$, which is the desired effect of $\exp(iH_C\theta)$. So suppose we have a parity network with a small size, we can implement the operation $\exp(iH_C\theta)$ by appropriately inserting R_z gates where a new binary term is generated. See fig. 6 for an example of how the parity network in fig. 1b can be transformed into a circuit implementing $\exp(iH_C\theta)$.

A CNOT gate can also be represented as an invertible matrix over $\mathbb{F}_2^{n\times n}$, i.e.,

$$\text{CNOT}(i,j) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & 1 & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}.$$

Thus the transformation applied by an n-qubit CNOT circuit can be written as an invertible matrix $\mathbb{M} \in \mathbb{F}_2^{n\times n}$, and the optimization of CNOT circuit is equivalent to transforming $\mathbb{M}$ to identity using row-addition operations, and each row-addition actually corresponds to a CNOT gate. Therefore, the optimization of CNOT circuit is equivalent to minimizing the number of row-additions to transform an invertible matrix to identity.

B Proof of theorem 2.1

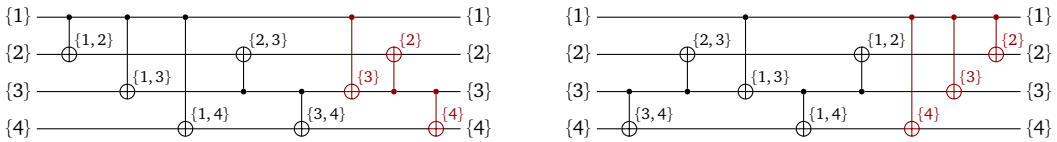
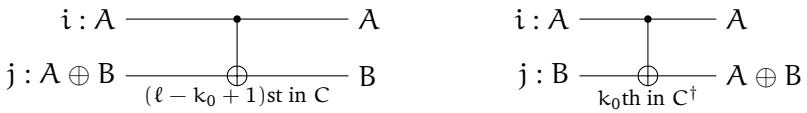


Figure 7: (a) The parity network fig. 1b and (b) its inverse.

We prove by induction on the number of gates executed in the circuit. To be precise, given a parity network C of ℓ operations and its inverse $C^\dagger$, for all $k = 0, 1, 2, \ldots, \ell$, we prove that if we execute the first k operations in $C^\dagger$ and the first $\ell - k$ operations in C, the terms on wire i in C will be the same as the term on wire i in $C^\dagger$.

Base case: When $k = 0$, the statement is trivially true, since the C has finished with the term on wire i in C being $\{i\}$, and $C^\dagger$ has not started yet with the term on wire i being $\{i\}$.

Inductive step: Suppose the statement is true for $k = 0, 1, \ldots, k_0 - 1$. We suppose the $(\ell - k_0)$-th operation in C is $\textsc{cnot}(i, j)$ and thus makes the k_0-th operation in $C^\dagger$ being $\textsc{cnot}(i, j)$. By the induction hypothesis, the terms on wire i in C and i in $C^\dagger$ are the same, and we denote them by A. Likewise, we use B to denote the term on j. So after the k_0-th operation, the wire j in $C^\dagger$ evaluates to $A \oplus B$, the same as the term on wire j before the $(\ell - k_0 + 1)$-th operation in C. Therefore the statement is true for k_0 and the induction is complete.



By the induction, every term generated in C has also been generated in $C^\dagger$ (and vice versa) and theorem 2.1 is proved.

C Proof of theorem 2.4

For an integer $k \geqslant 2$, a *finite projective plane* of order k is a plane with $k^2 + k + 1$ points and $k^2 + k + 1$ lines such that

(P1) $k + 1$ points on each line;

(P2) $k + 1$ lines through each point;

(P3) for any two distinct points, there is a unique line passing through them; and

(P4) for any two distinct lines, there is a unique point on both lines.

Theorem C.1 ([19])**.** *For each a prime power* k*, there exists a finite projective plane of order* k*.*

We give the following lemma which is used in our construction, whose proof is deferred to the end of this section.

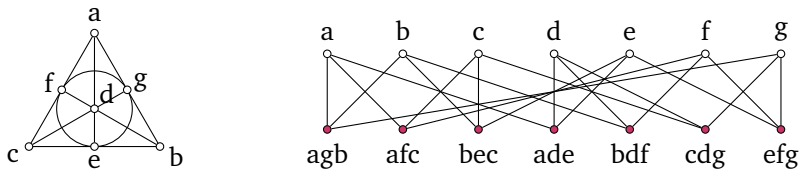
Lemma C.2. *For any* $t \geqslant 21$*, the largest prime number* p *satisfying*

$$p^2 + p + 1 \leqslant t \tag{6}$$

also satisfies

$$p^2 + p + 1 > \frac{t}{4}.$$

We construct a bipartite graph out of a finite projective plane as follows; see Figure 8 for an illustration.



(a) (b)

Figure 8: (a) The Fano plane, a finite projective plane of order 2, and (b) the bipartite graph constructed.

Proof. For arbitrary $n \geqslant 42$, we take k to be the largest prime power such that $k^2+k+1 \leqslant n/2$, and take a finite projective plane of order k. We construct a graph G as follows. For each point p and each line ℓ in the projective plane, we introduce a vertex, denoted by u_p and v_ℓ, respectively. We add an edge between u_p and v_ℓ if and only if p is on ℓ. Finally, we add $n-2(k^2+k+1)$ isolated vertices to make n vertices in total. By (P1), (P2) and theorem C.2, $m = (k^2+k+1)(k+1) = \Theta\left(n\sqrt{n}\right)$. The graph cannot contain any triangle because every edge is between a point vertex and a line vertex; moreover, by (P3) and (P4), there cannot be an induced cycle on four vertices. Thus, the girth of G is at least five. □

Finally, to prove theorem C.2, we restate a classical result from elementary number theory:

Theorem C.3 (Bertrand's Postulate [8])**.** *For any integer $n > 1$, there exists a prime p such that $n < p < 2n$.*

The proof of theorem C.2 proceeds as follows:

Proof. Let x be the largest positive integer satisfying $x^2+x+1 \leqslant t$. When $t \geqslant 21$, x must exist and $x \geqslant 4$. This implies $\left\lceil \frac{x}{2} \right\rceil \geqslant 2$. By theorem C.3, there exists a prime q such that

$$\left\lceil \frac{x}{2} \right\rceil < q < 2\left\lceil \frac{x}{2} \right\rceil \leqslant x+1,$$

i.e.,

$$\left\lceil \frac{x}{2} \right\rceil < q \leqslant x.$$

Since p is the largest prime satisfying (6),

$$p \geqslant q > \left\lceil \frac{x}{2} \right\rceil,$$

which yields

$$p^2+p+1 \geqslant \left\lceil \frac{x}{2} \right\rceil^2 + \left\lceil \frac{x}{2} \right\rceil + 1 > \frac{t}{4}.$$

□

D On non-biconnected perfect cancellation graphs

Observation D.1. *If G_1, G_2 are perpane graphs with perfect parity networks C_1, C_2, and they share a common vertex v, when the vertex of G_1, G_2 at v is perpane, with a perfect parity network constructed by simply concatenating C_1 and C_2.*

The proof is straightforward: $|C_1 \cup C_2| = |C_1| + |C_2| = |V_1| + |E_1| - 1 + |V_2| + |E_2| - 1 = |V_1 \cup V_2| + |E_1 \cup E_2| - 1 = |V| + |E| - 1$.

Theorem D.1 suggests when coming accross a non-biconnected graph, we should first decompose it into biconnected components, and then synthesize a graphic parity network for each component. The final circuit is obtained by concatenating the circuits for each component. With a perfect cancellation ordering for the entire graph, we can easily obtain a perfect cancellation ordering for each component as follows:

Lemma D.2. *Given a perfect cancellation ordering σ of a graph G and a biconnected component G_i of G, then $\sigma' := \sigma|_{V(G_i)}$ is a perfect cancellation ordering of G_i.*

Proof. For a non-cut vertex v in G_i, $N(v) \subset V(G_i)$, so $N^+_{\sigma'}(v) = N^+_{\sigma}(v)$ and remains σ'-linked. For a cut vertex v, let C be the component of $G - v$ that contains $G_i - v$, then $(N(v) \cap C) \subset G_i - v$, so $N^+_{\sigma}(v) \cap C = N^+_{\sigma'}(v) \cap C$ remains σ'-linked. Therefore, σ' is a perfect cancellation ordering of G_i. □

Finally, theorem 1.4 is a direct consequence of theorem 2.5, theorem D.1 and theorem D.2, since all cut vertices and biconnected components can be obtained in linear time [16].

E Proof of Theorem 1.3

Proof of Theorem 1.3. We have seen in Theorem 2.6 that Algorithm 2 always correctly synthesizes a graphic parity network. We now analyze its expected size. Each operation in lines 6, 14, and 18 generates a term for a new edge. Thus, the total number of them is m. Line 8 is executed at most once for each vertex, and hence the total number is at most n. Let s_{12} denote the expected number of executions of line 12. Therefore, the expected circuit size synthesized by this algorithm is at most $m + 2s_{12} + n$, which is $m + O(n^{1.5}\sqrt{2\log n})$ because

$$\begin{aligned}\min_{1\leqslant t\leqslant n}\left\{2\sum_{i<t} d_i + \frac{(n-t)n\log n}{d_t}\right\} &\leqslant \min_{1\leqslant t\leqslant n}\left\{2td_t + \frac{(n-t)n\log n}{d_t}\right\}\\ &\leqslant \min_{d}\left\{2nd + \frac{n^2\log n}{d}\right\} = n^{1.5}\sqrt{2\log n}.\end{aligned}$$

Moreover, when all but a constant number c_1 of vertices have degrees at least $c_2 n$, we have

$$\min_{1\leqslant t\leqslant n}\left\{2\sum_{i<t} d_i + \frac{(n-t)n\log n}{d_t}\right\} \leqslant 2\sum_{d_i<c_2 n} d_i + \frac{(n-c_1)n\log n}{c_2 n} \leqslant 2c_1c_2 n + \frac{1}{c_2}n\log n.$$

Thus, the size is $m + O(n\log n)$. □

F Sketch of the parameterized algorithm

First we give the definition of nice tree decomposition. A tree decomposition $\mathcal{T} = (T, \{X_t\}_{t\in V(T)})$ is *nice* if it satisfies the following properties:

- $X_r = \emptyset$ and $X_\ell = \emptyset$ for every leaf ℓ of T. In other words, all the leaves as well as the root contain empty bags.
- Every non-leaf node of T is of one of the following three types:
  - Introduce node: a node t with exactly one child t' such that $X_t = X_{t'} \cup \{v\}$ for some vertex $v \notin X_{t'}$; we say that v is *introduced* at t.
  - Forget node: a node t with exactly one child t' such that $X_t = X_{t'} \setminus \{v\}$ for some vertex $v \in X_{t'}$; we say that v is *forgotten* at t.
  - Join node: a node t with two children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$.

Now we introduce the definition of canonical representation which helps us to compress the solution space. For a node $t \in V(T)$ and a permutation σ of V_t and a vertex $v \in X_t$, the *canonical representation* of $\sigma|_{N^+_\sigma(v)}$ in σ under X_t is an ordered set of endpoints $\{(p_1, q_1), (p_2, q_2), \ldots, (p_k, q_k)\}$ such that for all i,

1. $p_i, q_i \in X_t \cup \{\dashv, \vdash\}$, only p_1 is allowed to be $\dashv$, only q_k is allowed to be $\vdash$,
2. segments do not intersect, i.e., $\sigma(p_i) < \sigma(q_i) < \sigma(p_{i+1})$,
3. the subset of elements in $N^+_\sigma(v)$ covered by this segment, i.e., $C_i = \{v \mid v \in N^+_\sigma(v) \wedge \sigma(p_i) \leqslant \sigma(v) \leqslant \sigma(q_i)\}$, is σ-linked,
4. there are no two consecutive X_t elements in $\sigma|_{C_i}$, and
5. all segments together cover all $N^+_\sigma(v)$, i.e., $\bigcup_{i=1}^{k} C_i \setminus \{\dashv, \vdash\} = N^+_\sigma(v)$.

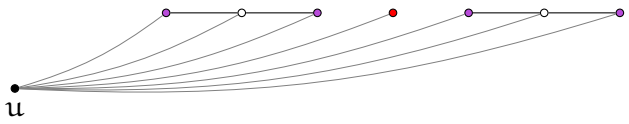
In our algorithm, when working from the leaves to the root, we will keep track of all canonical representations of $\sigma|_{N^+_\sigma(v)}$ for $v \in X_t$. To be precise, for each node $t \in V(T)$, we will construct a set c_t that consists of all pairs (τ, b) describing a valid order σ such that:

(V1) τ is the restriction of σ to X_t, and

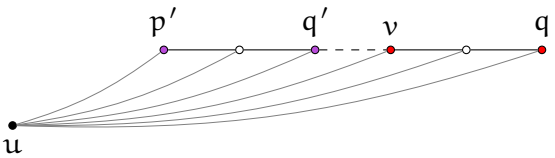
(V2) for each v in the current bag, b_v is the canonical representation of $\sigma|_{N^+_\sigma(v)}$ under X_t.

Now we discuss how c_t can be constructed in a bottom-up manner, and finally the graph is a perfect cancellation graph if and only if for the root bag r, $c_r \neq \emptyset$. For leaf nodes, the construction is trivial.

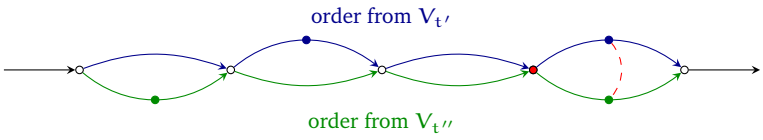
For an introduce node t with child t' such that $X_t = X_{t'} \cup \{v\}$ for some v, we process all canonical representations τ', b' in c'_t. We enumerate the position in τ' after which v is being inserted, and for a vertex $u \in N(v)$ that precedes v, its succeeding neighbors will also add v. Then we check whether v is being placed between two segments in b'_u since only the endpoints of the segments are allowed to have new neighbors, and if all checks pass, we can construct b_u by inserting (v, v) to the appropriate position in b'_u.



For a forget node t with child t' such that $X_t = X_{t'} \setminus \{v\}$ for some v, we process all canonical representations τ', b' in c'_t. We first check whether the succeeding neighbors of v are connected, and if not, since no new neighbors of v will be introduced, $N^+_\sigma(v)$ will never become σ-linked, we skip this τ', b'. Then we construct τ, b by deleting v from τ' and b'. For each $u \in N(v)$ that lies in the front of v, we check if v can be deleted safely from its succeeding neighbors. Without loss of generality, we assume there is a segment (v, q) in b'_u, and the predecessor of (v, q) is (p', q'), then we check whether q' is connected to v (the dashed edge must exist), because if not, after v is forgotten, q' can never be connected to v so $N^+_\sigma(u)$ can never be σ-linked. If the check passes, we should replace the segments $(p', q'), (v, q)$ with a single (p', q), and b_u is constructed.



For a join node t with children t', t'', then two valid orders in the two subtrees $V_{t'}$ and $V_{t''}$ can be merged only if between any pair of adjacent X_t vertices in the order, either all vertices are from $V_{t'}$ or all vertices are from $V_{t''}$, otherwise the endpoint after which clash occurs (indicated red below) will not have a σ-linked succeeding neighborhood. On the other hand, once the condition holds, the full order can be constructed by arbitrarily merging the two orders, and b_u can be constructed by simply merging b'_u and b''_u as well.


order from $V_{t'}$
order from $V_{t''}$