Optimized synthesis of 2-degree parity network

Yiren Lu
State Key Lab of Processors, Institute of Computing Technology, CAS, Beijing 100190, China
School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China

Junhong Nie
State Key Lab of Processors, Institute of Computing Technology, CAS, Beijing 100190, China
School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China

Xiaoming Sun
State Key Lab of Processors, Institute of Computing Technology, CAS, Beijing 100190, China
School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China

Guojing Tian
State Key Lab of Processors, Institute of Computing Technology, CAS, Beijing 100190, China
School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract

We study the problem of synthesizing 2-degree parity network. The problem takes a graph of n vertices and m edges as input, and outputs a sequence of row-addition operations that, when applied one-by-one to an identity matrix of size n, $\forall (u, v) \in E$, there exists a moment where there is a row vector $\mathbf{r}$ in the matrix with $\mathbf{r}_u = \mathbf{r}_v = 1$ and $\mathbf{r}_i = 0$ for all $i \neq u, v$, and finally transform the matrix back to identity.

We prove a lower bound $m + n - 1$ of the size (number of row-addition operations) of the parity network for a graph with m edges and n vertices. And we propose an optimal algorithm for constructing parity networks with sizes matching the lower bound for a special class of graphs, namely chordal graphs. Furthermore, we construct a family of graphs whose parity network requires $m + \Omega\left(n\sqrt{n}\right)$ operations, demonstrating the impossibility of a good solution on relatively sparse graphs. A simple randomized algorithm that applies to general graphs is given, with performance $m + \widetilde{O}\left(n\sqrt{n}\right)$ matching the worse-case lower bound, thus is tight (up to a logarithmic factor).

The 2-degree parity network is a family of quantum circuits that is a widely used component in many quantum algorithms for combinatorial optimization problems with Ising formulation, such as Max-Cut, and minimizing the size of the parity network means reducing the number of quantum operations in the quantum circuit. Thus our result will be of great help in the current noisy intermediate-scale quantum (NISQ) era, where the noise rate of quantum computing devices is high and the circuit size should be minimized to provide higher fidelity.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Mathematics of computing → Probabilistic algorithms; Computer systems organization → Quantum computing

Keywords and phrases Quantum computation, quantum circuit optimization, parity network, chordal graph, randomized algorithm

1 Introduction

Recently we are seeing emerging progress in the field of quantum computation, with the great development both experimentally and theoretically [5, 12, 13, 6]. As the scale of quantum computers goes up to hundreds to thousands of qubits, more efforts should be made to unleash the full power of quantum computers and chase the potential quantum advantage over classical computation. However, in this noisy-intermediate-scale quantum (NISQ) era [16], the noise rate of quantum computers is quite high, which means if the circuit size is too large, then the noise will accumulate and overwhelm any meaningful computational results. So given a specific computing task, designing a smaller circuit means not only faster computation but also less accumulated noise and more reliable results.

In this NISQ era, researchers have proposed many quantum algorithms which are believed to have some robustness against noise, such as variational quantum eigensolver (VQE) [15] and quantum approximate optimization algorithm (QAOA) [8], the former of which is used to find the eigenvalues of Hamiltonians that arise in many physics problems and the latter is utilized to tackle combinatorial optimization problems of interest in the computer science community. These algorithms will serve as the pioneers of quantum computation in the near future, making optimizing the circuit size of such algorithms more urgent and significant.

Our efforts focus on optimizing the circuit size of parity networks, which is a family of quantum circuits with some properties (Definition 2) first introduced by Amy *et al.* in [1]. Parity networks are important because they are widely-used gadgets in many quantum algorithms, such as quantum adiabatic algorithm (QAA) [9], QAOA [8], Hamiltonian simulation, etc. Therefore, optimizing the size of parity networks leads to a smaller quantum circuit size for many useful algorithms, making these quantum algorithms more likely to be realized in NISQ devices. Many *heuristic* attempts have been made to synthesize smaller parity networks [1, 10, 7], but no theoretical analysis for optimality or performance is known. Now we give the formal definition of the problem.

▶ **Definition 1 (Characteristic vector).** *Given a set $S \subset [n]$, the characteristic vector of S is an n-ary vector χ_S defined by $\chi_{S,j} = 1$ if $j \in S$ and 0 otherwise. Moreover, we abbreviate $\chi_{\{e\}}$ as χ_e and even χ_S as S if there is no ambiguity.*

▶ **Definition 2 (Parity network[1, Definition 2.3]).** *A parity network for $U \subset 2^{[n]}$ is a sequence of row-addition operations $R_1, \ldots, R_m \in \mathbb{F}_2^{n\times n}$ such that*

1. $R_m R_{m-1} \cdots R_1 = I$,
2. $\forall S \in U, \exists i \in [m], j \in [n]$, *such that* $(R_i R_{i-1} \cdots R_1)_{j,\cdot} = \chi_S$.

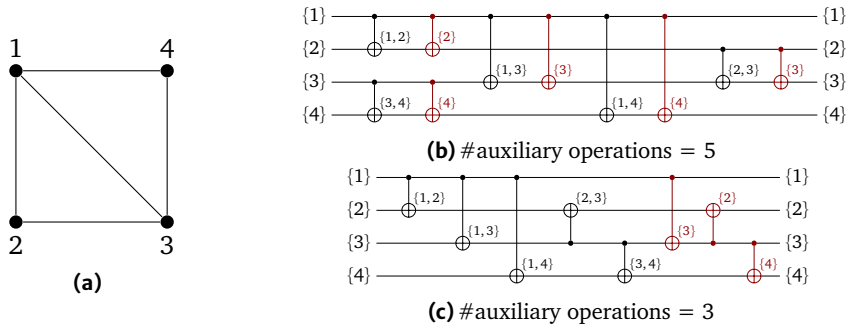
We denote the row-addition operation that adds the ith row to the jth row by $\textsc{Add}(i, j)$, and we will refer to j as the *target row*. If after a certain $\textsc{Add}(i, j)$, row j becomes χ_S, then we say a *term* S is generated on row j by this operation. From this aspect, a parity network is a sequence of operation that generates all terms $S \in U$.

In this paper, we consider a special yet important type of parity network with $\forall S \in U, |S| = 2$, reformulated as follows:

▶ **Definition 3.** *A 2-degree parity network for $G = (V, E)$ is a sequence of row-addition operations $R_1, \ldots, R_m \in \mathbb{F}_2^{n\times n}$ such that*

1. $R_m R_{m-1} \cdots R_1 = I$,
2. $\forall (u, v) \in E, \exists i \in [m], j \in [n]$, *such that* $(R_i R_{i-1} \cdots R_1)_{j,\cdot} = \chi_{\{u,v\}}$.

See Figure 1 for an example of two possible parity networks for a graph.



(a)

(b) #auxiliary operations = 5

(c) #auxiliary operations = 3

Figure 1 An example of two possible parity networks **(b,c)** for a graph **(a)**. In diagram **(b,c)**, the ith line from bottom to top denotes the value of row i. The $\bullet\!\!-\!\!\oplus$ symbol signifies a row-addition operation: if $\bullet$ is on the ith line, and $\oplus$ is on the jth line, it represents $\textsc{Add}(i,j)$. The annotations S on the target rows indicate their values are χ_S after executing the operation operations, and the auxiliary operations are colored red. Clearly, **(c)** has a smaller size. It will be later proved in Theorem 4 that for this graph with 4 vertices and 5 edges, the lower one with 8 operations is the optimal parity network.

The 2-degree parity networks that we considered are used in QAA [9] and QAOA [8] mentioned above when applied to solve 2-degree combinatorial optimization problems such as Max-Cut:

$$\max_{x\in\{-1,1\}^n} \sum_{(u,v)\in E} \frac{1-x_u x_v}{2}, \tag{1}$$

where n is the graph size and E is the set of edges, and may even be useful for higher degree problems if they include 2-degree terms. Solving thse problems with QAA and QAOA has drawn much attention in the quantum computing community [20, 21, 22, 23, 4, 3, 2, 18], showing that it is a quite important and popular topic. In fact, there have been preliminary physical demonstrations of these quantum algorithms on NISQ devices [14, 11]. The popularity of solving 2-degree combinatorial optimization problems with QAA and QAOA makes the synthesis of 2-degree parity networks a quite fruitful topic. Besides the significance of the problem to quantum computation, the parity network is a reversible boolean circuit describing a type of elimination process defined on $\mathbb{F}_2$ matrices, which might be of independent interest in other fields. In addition, 2-degree parity networks can be seen as a structure associated with graphs and synthesizing 2-degree parity networks can be seen as a succinct and interesting graph problem on its own, and in fact our result is closely related to graph theory.

Our problem is formally defined as follows:

Problem:	MINIMUM 2-DEGREE PARITY NETWORK
Input:	A connected undirected graph $G = (V, E)$.
Output:	A parity network (c.f. Definition 2) for G with the minimum size.

Note that for simplicity, we omit the *2-degree* in the remaining text, as we will only consider 2-degree parity networks in this manuscript. A trivial upper bound of the size is $2m (m = |E|)$, as seen in Figure 1(b) that there is always a naive construction where every term takes 2 operations. A trivial lower bound is m as each operation can only generate one term. Therefore, minimizing the size is equivalent to minimizing the number of operations that does not generate any new desired term, and we call them *auxiliary operations*, which is colored red in Figure 1. Moreover, we define $\mathcal{H}(G)$ to be the minimum number of auxiliary

operations in a parity network for G, and trivially we have

$$0 \leqslant \mathcal{H}(G) \leqslant m. \tag{2}$$

Remark that if we can find a solution with close to zero auxiliary operations, then we will be able to cut down the number of opeartions by half with respect to the naive method, which is a significant improvement and will be of great help in the NISQ era.

Throughout the manuscript, we will always refer to $n = |V|$ as the number of vertices in the graph, and $m = |E|$ as the number of edges. Moreover, if a graph is not connected, then we can independently synthesize the parity network for each of its connected components. Therefore, in this manuscript, we will always assume **the graph is connected**.

Our contributions

Upon defining the problem, the first natural question that arises is whether we can establish a lower bound on the parity network size, and then design algorithms to build parity networks that match that size. Following the route, in Section 3.1, we first give the following result:

▶ **Theorem 4.** *For an arbitrary graph $G = (V, E)$, a parity network for G contains at least $m + n - 1$ operations, i.e., $\mathcal{H}(G) \geqslant n - 1$.*

This theorem says the best parity network for a graph will be composed of m operations generating all the desired terms and $n - 1$ auxiliary operations. The proof of this theorem gives some further structural information about parity networks, which will be discussed in detail in Section 3.1. We refer to a parity network that fits the lower bound as an *ideal* parity network.

If a graph admits an ideal parity network, we call it *parity-traversable*. Not all graphs are parity-traversable.What is more, unfortunately, there do exist graphs with significant $\mathcal{H}(G)$. Here comes our second result, presented in Section 3.2:

▶ **Theorem 5.** *There exists a graph family $\{G_n\}_{n\in S}$ such that $\mathcal{H}(G_n) = \Omega\left(n\sqrt{n}\right)$, where S is an infinite set of integers.*

We give an explicit construction of the graph family via a delicate structure taken from geometry theory, namely the projective plane, and prove the graph family is hard for MINIMUM 2-DEGREE PARITY NETWORK. The large gap between $n - 1$ and $n\sqrt{n}$ suggests that there might not be a universal procedure to synthesize a good parity network for all graphs, and that we must exploit the structural properties of the input graph to do better.

Our next result is on parity-traversable graphs and how to synthesize an ideal parity network for them. We have the following results in Section 4:

▶ **Theorem 6.** *Chordal graphs are parity-traversable. Moreover, there is a linear-time algorithm that synthesizes an ideal parity network for a chordal graph.*

The theorem is proved using a simple and elegant construction that builds the ideal parity network for a chordal graph. The algorithm leverages the properties of perfect elimination ordering of chordal graphs. However, we have discovered that there are graphs that are not chordal but still compatible with our synthesis algorithm, hence admitting ideal parity networks, namely,

▶ **Proposition 7.** *There are graphs that are not chordal but parity-traversable.*

It is an interesting problem to determine exactly which graphs admit ideal parity networks. We discuss our preliminary results on this problem in Appendix A and leave it as an open problem.

Besides synthesizing the ideal parity network for parity-traversable graphs, the most advantageous for us to work with, another important problem remains how well we can do for general graphs without strong structural assumptions such as chordality? Therefore, in Section 5 we propose a randomized linear-time algorithm with a rigorous performance guarantee and analyze the expected size in the parity network it synthesizes. The number is only related to the vertex degrees of the input graph:

▶ **Theorem 8.** *For a graph* $G = (V, E)$ *with degrees of vertices sorted in ascending order* $d_1, d_2, \ldots, d_n$*, there is a linear-time algorithm that synthesizes a parity network for* G *with expected size*

$$m + O\left(\min_{1\leqslant k\leqslant n}\left\{\sum_{i\leqslant k} d_i + \frac{(n-k)n\log n}{d_{k+1}}\right\}\right). \tag{3}$$

The bound above is quite complicated, and we present corollaries from it for readers to better understand the result.

▶ **Corollary 9.** *For a graph* $G = (V, E)$*, we have:*

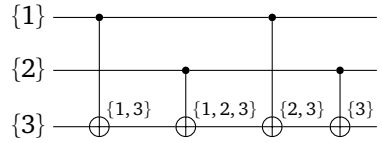
1. *There is an algorithm synthesizing a parity network for* G *with expected size* $m + \widetilde{O}\left(n^{1.5}\right)$.
2. *If each vertex has degree* $\Omega(n)$*, there is an algorithm synthesizing a parity network for* G *with expected size* $m + \widetilde{O}(n)$.
3. *If* G *has maximum and minimum degree* $\Delta(G)$ *and* $\delta(G)$*, there is an algorithm synthesizing a parity network for* G *with expected size* $m + O\left(n \cdot \min\left\{\Delta(G), \frac{n\log n}{\delta(G)}\right\}\right)$.

Note that as pointed out by Theorem 5, there are hard instances for MINIMUM 2-DEGREE PARITY NETWORK with $\mathcal{H}(G) = \Omega\left(n\sqrt{n}\right)$, so the bound given by our algorithm (Corollary 9, Item 1) is tight (up to a polylogarithmic factor) considering the worst-case scenario. Moreover, as seen in Corollary 9, Item 2, our algorithm is also tight for a case of dense graphs. It is an important problem for future work to determine $\mathcal{H}(G)$ for more general input graphs and design better algorithms to close the gap between the upper and lower bounds.

The proof of Theorem 8 is given in Section 5. This result does not tell us the solution quality with respect to the optimal parity network for a graph, but it tells us a denser graph is more beneficial for optimization, which is intuitively reasonable. For example, if a graph is dense with all vertices of degree $\Omega(n)$ as specified in Corollary 9, Item 2, then we will cut down the size by a factor of $\left(1 - \frac{\log n}{n}\right)/2$ compared to the trivial method of $2m$ operations.

Discussion and open problems

Though we have proved several lower bounds and proposed synthesizing algorithms, we still do not have a clear understanding of what an optimal parity network must look like, and one of the most curious problems on this topic is whether terms of size $\geqslant 3$ as intermediate process are necessary. We have the following example showing that parity networks with size 3 terms could possibly be ideal (hence optimal),



but we have no evidence showing there is an optimal 2-degree parity network that **must** introduce terms of size $\geqslant 3$. We propose the following conjecture that requires further investigation:

▶ **Conjecture 10.** *If* G *has a parity network of* k *operations, there is a parity network for* G *of* k *operations that only generates terms of size* $\leqslant 2$.

The result Proposition 7 opens another direction for future work, that is to give parity-traversable graphs an explicit characterization. Little is known about parity-traversable graphs besides the following preliminary result:

▶ **Proposition 11.** *Parity-traversable graphs do not forbid any subgraph.*

Looking through the known superclasses of chordal graphs, we are currently unable to find a match, so we conjecture parity-traversable graphs might be a previously unknown graph class, so characterising it is an exciting graph theory problem. Moreover, if we can find such a characterization, we can define PT-COMPLETION, similar to CHORDAL-COMPLETION, to be the problem of adding a minimum number of edges to make a graph parity-traversable. If we have a good solution to PT-COMPLETION, then for arbitrary graphs, we can first complete them to parity-traversable graphs and then synthesize the ideal parity network, which intuitively could beat the randomized algorithm on general graphs which merely considers the degrees.

Our work deals with a restricted version of MINIMUM PARITY NETWORK. So in future work it remains to generalize or propose new theoretical results on solving MINIMUM PARITY NETWORK. Another problem that has been opened for years is to determine the hardness of MINIMUM PARITY NETWORK. [1] shows the NP-hardness of two variants of it, but no hardness result for the original problem is known. Proving the NP-hardness of MINIMUM 2-DEGREE PARITY NETWORK seems a promising strategy to prove the NP-hardness MINIMUM PARITY NETWORK.

2 Preliminaries

Chordal graphs

We define chordal graphs, a family of graphs family that is useful throughout the manuscript.

▶ **Definition 12** (Chordal graph). *A graph is chordal if every cycle of length* $\geqslant 4$ *has a chord, that is, an edge connecting two non-adjacent vertices on the cycle. An equivalent definition is that a graph is chordal if it has no induced cycle of length* $\geqslant 4$.

▶ **Definition 13** (Neighbors in a sequence). *Given a graph* $G = (V, E)$*, a permutation* $\pi \in \mathrm{Sym}(V)$*,* $\forall i, 1 \leqslant i \leqslant n$*, we define two ordered set of indices:*

1. $\Gamma_i^+ = \{j \mid (\pi_j \in N(\pi_i)) \wedge (j > i)\}$*, i.e., the indices of all neighbors of* π_i *with indices larger than* i.
2. $\Gamma_i^- = \{j \mid (\pi_j \in N(\pi_i)) \wedge (j < i)\}$.

▶ **Definition 14** (Perfect elimination order). *Given a graph* $G = (V, E)$*, a permutation* $\pi \in \mathrm{Sym}(V)$ *is a perfect elimination order if for each* i*,* Γ_i^+ *forms a clique in* G.

It is known that a graph G has a perfect elimination order if and only if G is chordal [17, Theorem 1]. Moreover, if a graph is chordal, a perfect elimination order can be constructed in linear time [19].

Finite projective plane

Now we define finite projective planes, a geometry structure used to construct our hard instance in Section 3.2 to prove Theorem 5.

▶ **Definition 15 (Finite projective plane).** *A finite projective plane (FPP) of order* $k(k \geqslant 2)$ *is a plane with:*

1. $k^2 + k + 1$ *points,*
2. $k^2 + k + 1$ *lines,*
3. $k + 1$ *points on each line,*
4. $k + 1$ *lines through each point.*

Moreover,

1. *For any two distinct points, there is a unique line passing through them.*
2. *For any two distinct lines, there is a unique point on both lines.*

▶ **Theorem 16.** *There exists an FPP of order* k *if* k *is a prime power.*

3 Lower bound on the number of auxiliary operations

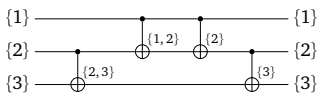
In this section, we first establish a lower bound on the number of auxiliary operations in a parity network in Theorem 4, which applies to arbitrary graphs. Then we construct a family of hard instances and prove the stronger lower bound in Theorem 5.

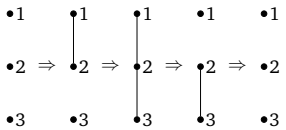
3.1 Lower bound for general instances

This lower bound will guide the design of our algorithms in the following sections. Actually, we will prove a stronger version of the theorem, which is Lemma 17.

▶ **Lemma 17.** *For an arbitrary parity network* C *with* ℓ *operations that synthesizes* m *terms of size* 2 *while the terms form a connected graph of size* n*, we assert that there must be at least* $n - 1$ *operations that produce terms with size* 1 *or* $\geqslant 3$*.*

Clearly, Theorem 4 is a direct consequence of Lemma 17. Now we give some intuition behind the proof of Lemma 17, informally proving the lemma for parity networks that *only generate terms of size* 1 *and* 2 as a warmup. After the execution of the ith operation, there will be some terms of size 2 on the rows. For a term $\{u, v\}$, we see it as an *edge* (u, v). If we look at the start and the end of the parity network, each row is a term of size 1, hence corresponding to a graph with n connected components. However, if we look at the graph formed using all the edges generated by the parity network, then the graph must be connected.



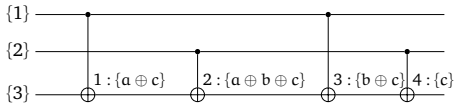


A possible way of achieving this might be as follows (as demonstrated in the diagram above): generate all the edges to make the graph connected, and then remove $n - 1$ edges and split the graph into n connected components. However, this might not be the case, since we can add and remove edges alternately. But by some sort of accounting method, we will find the number of edge removal is at least $n - 1$. It is easy to say if there are no size-$\geqslant 3$ terms in the parity network, each removal operation must correspond to a operation that generates a term of size 1, and thus the weakened lemma is proved. Now we present the complete proof.

The complete proof leverages the idea of keeping track of the connectivity determined by the terms and handles some corner cases that arise when we allow the parity network to generate terms of size $\geqslant 3$. Namely, in the warmup case, each *edge removal* operation that splits a connected component must generate a term of size 1, so it must be an auxiliary. However, this might not be the case in the general setting where terms of size $\geqslant 3$ are allowed. But fortunately, each such operation must corresponds to a (distinct) term of size 1 or $\geqslant 3$ as we will elaborate below.

Proof. Given S, a subset of the powerset 2^V of V, we define an equivalence relation R_S on V as follows:

$$R_S = \{(u, v) \mid \exists T \in S, u \in T \wedge v \in T\}. \tag{4}$$



$P_3 = \{\{a\}, \{b\}, \{b, c\}\}$
$H_3 = \{\{a\}, \{b\}, \{c\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$

Consider the execution process of the parity network. After executing the ith operation, for row j, suppose it now equals $\chi_{W_{i,j}}$, and we define $P_i = \{W_{i,j} \mid j \in [n]\}$ to be the set of terms on the rows after executing the ith operation, and let $H_i = \bigcup_{k=1}^{i} P_k$. An illustrative example is given above. Recall that there are ℓ operations in this parity network. We assert that $|V/R_{H_\ell}|$, the number of equivalence classes in V with respect to R_{H_ℓ}, is 1, as we must have generated all terms corresponding to all edges in the connected graph after executing ℓ operations, and all elements in V are equivalent to each other via those size-2 terms.

Then we analyze the role of the $(i+1)$th operation to H_i. Two events may happen to the number of equivalence classes in V with respect to R_{H_i} after executing an operation:

(H1) Decreased by 1. Namely, it makes two equivalence classes in V with respect to R_{H_i} become equivalent.
(H2) Unchanged.

Then as we have argued that the $|V/R_{H_\ell}| = 1$, we must have at least $n-1$ operations of the first type, as initially $|V/R_{H_0}| = n$. Similarly, we analyze the role of the $(i+1)$th operation to P_i. Three events may happen to the number of equivalence classes in V with respect to R_{P_i} after executing an operation:

(P1) Decreased by 1.
(P2) Increased by 1.
(P3) Unchanged.

Clearly, each (H1) operation must be a (P1) operation, as if an operation connects two equivalence classes for the first time in the history, it must connect two equivalence classes at the current time point. Then since finally each row is in a term of size 1 thus $|V/R_{P_\ell}| = n$, the number of (P2) operations must be at least $n-1$. Now we discuss the (P2) operations and argue that each (P2) operation corresponds to a operation that generates a parity term of size 1 or $\geqslant 3$. We have the following proposition on (P2) operations:

▶ **Proposition 18.** *A (P2) operation must perform* $\{A, B\} \to \{A, B \setminus A\}$ *for some* $A \subsetneqq B$.

Proof. Generally speaking, an operation might perform $\{A, B\} \to \{A, A \triangle B\}$, where $A \triangle B = (A \setminus B) \cup (B \setminus A)$ denoting the symmetric difference of A and B.

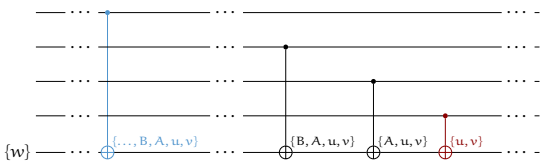
We denote the equivalence class of A before and after executing the operation by $[A]$, $[A]'$, respectively, and we know $[A] = [B]$. Then if $A \not\subset B$, then $A \setminus B \subset A \triangle B$, giving us

$$[A]' = [A \setminus B]' = [A \triangle B]', \tag{5}$$

thus $\forall a \in A, b \in B$, we still have $[a]' = [b]'$, so the number of equivalence classes does not increase. This means, for an operation to split an equivalence class, $A \setminus B$ must be $\emptyset$. ◀

Now we start to construct a set $\mathcal{A}$ of auxiliary operations in this network. For each (P2) operation g, if it generates a term of size 1 or $\geqslant 3$, then we directly add it to $\mathcal{A}$.

Otherwise, g generates a term of size 2, then it must perform $(A, A \cup \{u, v\}) \to (A, \{u, v\})$ (here we only show the value of the two rows involved in the operation). Suppose the target row of this operation is w, we identify the last non-(P2) operation targeting row w before g, and let the operation be h. We assert that such a operation must exist, as initially each row equals a term of size 1, and (P2) operations can only decrease the size of the parity term on w. Moreover, h must be an auxiliary operation, as the size of the term on w strictly decreases during the execution from operation h to operation g, and after executing g the size is 2. Therefore, we find a distinct corresponding auxiliary operation h (colored blue) for g (colored red) and add h to $\mathcal{A}$.



Following the analysis above, for each (P2) operation, we find a distinct correspondence in a set of auxiliary operations $\mathcal{A}$, which finally gives us $|\mathcal{A}| \geqslant n - 1$. ◀

3.2 Lower bound for a family of hard instances

This section gives the proof of Theorem 5. First, we present the construction of the hard instances. For any FPP of order t, we can construct a bipartite graph $G = (U, V, E)$ with $|U| = |V| = t^2 + t + 1$ and $|E| = (t^2 + t + 1)(t + 1)$ as follows:

1. Each point p in the projective plane corresponds to a vertex in U.
2. Each line ℓ in the projective plane corresponds to a vertex in V.
3. For each point p and line ℓ such that p is on ℓ, we add an edge (p, ℓ) to E.

Therefore, we have an infinite family of bipartite graphs $\{(U_n, V_n, E_n)\}_{n=p^{2k}+p^k+1, p\in\mathbb{P}}$ such that

1. $|E_n| = \Theta\left(n\sqrt{n}\right)$,
2. $\forall u_1, u_2 \in U_n, |N(u_1) \cap N(u_2)| = 1$,
3. $\forall v_1, v_2 \in V_n, |N(v_1) \cap N(v_2)| = 1$.

The diagram below gives the Fano plane, a finite projective plane of order 2 and the bipartite graph constructed from the Fano plane.

▶ **Lemma 19.** *For the graph family* $\{(U_n, V_n, E_n)\}_{n=p^{2k}+p^k+1, p\in\mathbb{P}}$ *defined above, we have*

$$\mathcal{H}(U_n, V_n, E_n) = \Omega\left(n\sqrt{n}\right). \tag{6}$$

Now we prove the lemma by establishing another lower bound on the number of operations of the parity network. For a parity network C for G, we can find all 2-literal terms generated in C, which we denote by $E^C \supseteq E$. For each $g = \textsc{Add}(u, v)$ that generates a desired term for

the first time, if we look at row v, there must be at least one operation targeting row v after g, because otherwise row v will not be restored to a term of size 1. We denote the first such operation by $s(g)$. Then we define the following two sets:

1. $\mathcal{F}$: the set of operations that generate a desired term for the first time.
2. $\mathcal{S}$: $\{s(g) \mid g \in \mathcal{F}\}$.

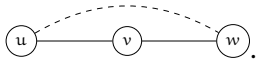
Note that $|\mathcal{F}| = |\mathcal{S}| = m$, and that

$$|C| \geqslant |\mathcal{F}| + |\mathcal{S}| - |\mathcal{F} \cap \mathcal{S}| = 2m - |\mathcal{F} \cap \mathcal{S}|. \tag{7}$$

Therefore, if we want to decrease $|C|$, we need to increase $|\mathcal{F} \cap \mathcal{S}|$. Each element in $\mathcal{F} \cap \mathcal{S}$ is a *fabricator* (operation that generates a desired parity term for the first time) right after another fabricator targeting the same row, which has to perform one of the following:

1. $(\{u, w\}, \{u, v\}) \to (\{u, w\}, \{w, v\})$ where $(u, v), (w, v) \in E$, and we denote the set of such operations by $\mathcal{U}_1$.
2. $(\{a, b, c, d\}, \{a, b\}) \to (\{a, b, c, d\}, \{c, d\})$ where $(a, b), (c, d) \in E$, and we denote the set of such operations by $\mathcal{U}_2$.

For the first case, the operation corresponds to a triangle in $\left(V, E^C\right)$:



Moreover, the triangle must contain at least two edges in E, giving us:

$$|\mathcal{U}_1| \leqslant \Delta_2\left(V, E, E^C\right). \tag{8}$$

where Δ_2 denotes the number of triangles in $\left(V, E^C\right)$ that contain at least two edges in E. We first show we cannot significantly decrease $|\mathcal{U}_1|$ without adding a significant number of edges to E^C, starting with the following simple observation:

▶ **Observation 20.** (U, V, E) *has no induced 4-cycle.*

This observation limits the number of triangles that can be *closed* by edges in $\left|E^C \setminus E\right|$, meaning that any edge in $E^C \setminus E$ can participate in at most *one single triangle* with two edges in E, thus increasing the $\Delta_2\left(V, E, E^C\right)$ by at most 1, i.e.,

$$\left|E^C \setminus E\right| \geqslant \Delta_2\left(V, E, E^C\right) \geqslant |\mathcal{U}_1|. \tag{9}$$

Now we look at the second case. For each $(\{a, b, c, d\}, \{a, b\}) \to (\{a, b, c, d\}, \{c, d\}) \in \mathcal{U}_2$, we have:

1. $(a, b), (c, d) \in E$
2. $\{a, b\}, \{c, d\}$ have never appeared.

If we generate a $\{a, b, c, d\}$, we assert that it might only help to increase $|\mathcal{U}_2|$ by 3, because there can be at most 3 pairs edges among $\{a, b, c, d\}$. So denote the number of size-4 terms by t, then

$$3t \geqslant |\mathcal{U}_2|. \tag{10}$$

Recall that $\mathcal{H}(U, V, E) \geqslant m - |\mathcal{F} \cap \mathcal{S}|$. Therefore, for $|\mathcal{F} \cap \mathcal{S}| = |\mathcal{U}_1| + |\mathcal{U}_2|$ to be $\Omega(m)$, we need $\left|E^C \setminus E\right| + 3t$ to be $\Omega(m)$, so this means $\mathcal{H}(U, V, E) = \Omega(m) = \Omega\left(n\sqrt{n}\right)$, and Lemma 19 is proved. Theorem 5 is thus proved as a direct consequence of Lemma 19.

Algorithm 1 Parity network synthesis for chordal graphs

```
1:  procedure SYTH-CHORDAL(G = (V, E), π)  ▷ A valid perfect elimination order of this graph.
2:      p ← [−1, −1, ..., −1]                ▷ Recording the state of each row.
3:      for i ← 1, 2, ..., n do
4:          for j ← i − 1, i − 2, ..., 1 do
5:              if (π_j, π_i) ∈ E then
6:                  if p_{π_j} = −1 then
7:                      ADD(π_i, π_j)
8:                  else
9:                      ADD(p_{π_j}, π_j)
10:                 p_{π_j} ← π_i
11:     RESTORE-SINGLE(p)
```

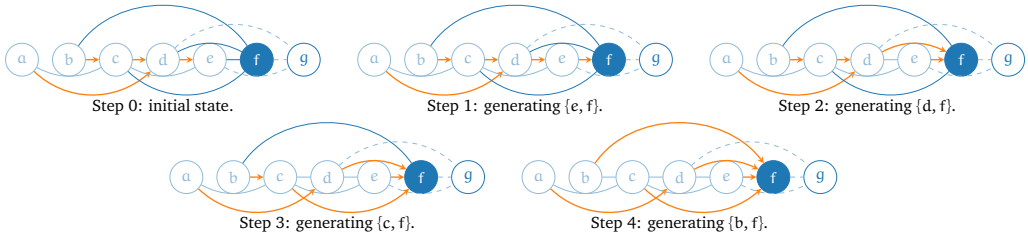


Figure 2 A working example of Algorithm 1. Suppose we begin to work on vertex f, and the solid edges are the ones we will now build, the faded edges are the ones built, the dashed edges are the ones to be built later, and orange arrows indicate the current state of each vertex, for example, in the first diagram row a equals $\chi_{\{a,d\}}$.

4 An optimal synthesis algorithm for chordal graphs

In this section, we give an algorithm that synthesizes a parity network for a chordal graph in linear time. We will first generate a perfect elimination ordering π for this graph, and then process this ordering one by one. When we are processing the ith vertex π_i in this ordering, we will generate all terms $\{\pi_j, \pi_i\}$ on row π_j, for all $j < i$, $(\pi_j, \pi_i) \in E$, and then after processing all vertices in π, we will have generated all desired terms in this graph. A demonstrative example can be seen in Figure 2, and the pseudocode of our algorithm is given in Algorithm 1.

▶ **Lemma 21.** *When given a chordal graph $G = (V, E)$ as input, then in Algorithm 1, after the execution of the i_0th loop in line 3, the following will hold:*

1. *row $\pi_j (j \in \{1, 2, \ldots, i_0 - 1\})$ will equal $\{\pi_j, \pi_i\}$ where π_i is the neighbor of π_j with the largest index i such that $i \leqslant i_0$ (or $\{\pi_j\}$ if there is no such neighbor),*
2. *row $\pi_j (j \in \{i_0 + 1, i_0 + 2, \ldots, n\})$ will equal π_j.*

Proof. We will prove the lemma by induction.

Base case: When $i_0 = 2$, the term on row π_1 is $\{\pi_1, \pi_2\}$ if $(\pi_1, \pi_2) \in E$, and $\{\pi_1\}$ otherwise. The term on row π_2 is $\{\pi_2\}$.

Inductive hypothesis: Suppose the lemma holds for $i_0 = k$.

Inductive step: Now we prove the lemma for $i_0 = k + 1$. We will look at the terms on corresponding rows in $\tau := \Gamma^-_{k+1}$. According to Algorithm 1, we will deal with these neighbors in a reversed order. Then for each π_ℓ $(\ell \in \tau)$,

1. If $p_{\pi_\ell} = -1$, then the term on row π_ℓ will be $\{\pi_\ell, \pi_{k+1}\}$ after line 7.

2. If $p_{\pi_\ell} = \pi_q$ for some $q \leqslant k$, then it means that q is the largest index such that $\ell < q \leqslant k$ and $(\pi_\ell, \pi_q) \in E$. However, since $(\pi_\ell, \pi_{k+1}) \in E$ and $(\pi_\ell, \pi_q) \in E$, we have $(\pi_q, \pi_{k+1}) \in E$ according to the definition of perfect elimination order. This means $q \in \tau$. Therefore, as $q > \ell$, we know the term on row π_q has already become $\{\pi_q, \pi_{k+1}\}$. Therefore, simply applying a $\textsc{Add}(\pi_q, \pi_\ell)$ as specified on line 9 will generate $\{\pi_\ell, \pi_{k+1}\}$ on row π_ℓ.

◀

The correctness of the algorithm will be clear from the proof of Lemma 21, and the size of the result circuit is $m + n - 1$. There is a linear-time algorithm that finds a perfect elimination order for a chordal graph [19]. And the time complexity of the first loop can be reduced to $O(n + m)$ using a finer implementation, finally completing the proof of Theorem 6.

5 A randomized algorithms for general graphs

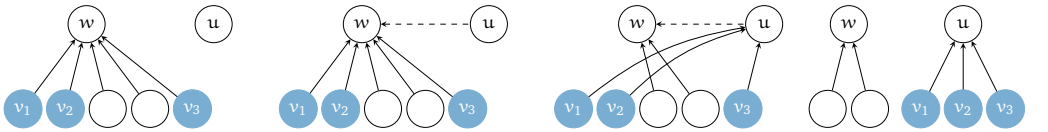
Algorithm 2 A randomized algorithm for parity network synthesis, where $S(x) = \{y \mid p_y = x\}$.

```
1:  procedure SYTH-RANDOMIZED(G = (V, E))
2:      π ← RANDOM-SHUFFLE(V)
3:      p ← [−1, −1, ..., −1]                          ▷ Recording the states.
4:      for i ← 1, 2, ..., n do
5:          if p_{π_i} ≠ −1 then
6:              for u ∈ S(p_{π_i}) ∩ N(π_i) do
7:                  ADD(π_i, u)
8:                  p_u ← π_i
9:                  E ← E \ {(u, π_i)}
10:             ADD(p_{π_i}, π_i)
11:             p_{π_i} ← −1
12:         for j ← 1, 2, ..., i − 1 do
13:             T ← S(π_j) ∩ N(π_i)
14:             if |T| ≠ ∅ then
15:                 ADD(π_j, π_i)                      ▷ A bridge.
16:                 for u ∈ T do
17:                     ADD(π_i, u)
18:                     p_u ← π_i
19:                     E ← E \ {(u, π_i)}
20:                 ADD(π_j, π_i)
21:         for u ∈ N(π_i) do
22:             if p_u ≠ π_i then
23:                 ADD(π_i, u)
24:                 p_u ← π_i
25:                 E ← E \ {(u, π_i)}
26:     RESTORE-SINGLE(p)
```

In this section, we give a randomized algorithm that synthesizes parity networks for general graphs with a rigorous analysis of its expected performance. It is simple and efficient, with no need for any preprocessing or postprocessing. The idea of the algorithm is as follows: suppose we process the vertices in a certain order, and when we are working on a certain vertex, we will generate all its related parity terms that have not shown up. Following this process, the values of all rows can be illustrated using a graph with directed edges $\{(v, u) \mid \text{row } v \text{ has term } \{u, v\}\}$, which is a forest of star graphs along with some isolated vertices.

When we work on vertex v, we need to find all $v \in N(u)$ such that $\{u, v\}$ have not been generated, and generate $\{u, v\}$ on row v. First we assume that u is one of the isolated vertices. There are two cases for $v \in N(u)$:

1. Row v equals $\{v\}$. Then v is an isolated vertex in the graph, and simply applying $\textsc{Add}(u, v)$ will do.
2. Row v equals $\{v, w\}$ for some w. These rows correspond to the leaves of the star graph and we need to build a *bridge* to help u *grab* v from w as follows: first apply $\textsc{Add}(w, u)$, after which the state of u will become $\{w, u\}$. Then apply $\textsc{Add}(u, v)$ for all v, through which the $\{v, w\}$ will be transformed to $\{v, u\}$. Finally we restore the state of u to $\{u\}$ by applying $\textsc{Add}(w, u)$ again.



Step 0: Initial state. Step 1: Building bridge. Step 2: Grabbing. Step 3: Removing bridge.

Finally, if u is currently $\{u, w\}$ for some w, then u should first grab its siblings – the neighbors v whose corresponding rows $\{v, w\}$. After that, row u can simply restore to a term of size 1 and then start the process described above.

Clearly, the circuit size generated by this algorithm is $m + O(n + K)$, where K is the number of bridges built. Now we will analyze $\mathbb{E}(K)$. We begin with a warm-up case where the graph is a dense graph with all vertices having degree $\geqslant cn$ for some constant c.

▶ **Lemma 22.** *For a graph $G = (V, E)$ and a random permutation $\pi \in \mathrm{Sym}(V)$, the expected number of bridges built by successor vertices to grab sons from $u = \pi_1$ is $O(\log n)$.*

The proof employs simple probability argument.

Proof. For each $v \in N(u)$, it will be grabbed from u as long as there comes the second neighbor of v in the random permutation. Suppose v has $c_v n (c_v \geqslant c)$ neighbors, and let X_v be position of the second neighbor of v in the random permutation minused by 1, then we have

$$\mathbb{P}(X_v = k) = \left(1 - \frac{c_v n - 1}{n-1}\right)\left(1 - \frac{c_v n - 1}{n-2}\right)\cdots\left(1 - \frac{c_v n - 1}{n-(k-1)}\right)\left(\frac{c_v n - 1}{n-k}\right) > (1-c_v)^{k-1} c_v, \tag{11}$$

which means

$$\mathbb{P}\left(X_v \leqslant \log_{\frac{1}{1-c}} n\right) \geqslant \mathbb{P}\left(X_v \leqslant \log_{\frac{1}{1-c_v}} n\right) > 1 - (1-c_v)^{\log_{\frac{1}{1-c_v}} n} = 1 - O\left(n^{-1}\right). \tag{12}$$

(12) tells us for a total of $\Theta(n)$ sons, the expected number of remaining sons of u is $O(1)$ after $\Omega(\log n)$ rounds, which means at most $O(1)$ more bridges are needed to grab all sons of u. Therefore, for u, we can assume all of the next $\Omega(\log n)$ vertices will build a bridge and after that only $O(1)$ additional bridges might be built and the lemma is thus proved. ◀

Now we have the cost of grabbing all sons of u when u is the first vertex, and now we consider the overall cost. Intuitively, after dealing with the first vertex and building all terms related to the first vertex, the remaining process can be seen as a recursion applied to the remaining graph without this vertex. And for each vertex in the remaining graph, fewer sons are needed to be grabbed from it vertex than if it is the first one in the sequence. Namely, we have the following lemma:

▶ **Lemma 23.**

$$\mathbb{E}(K) \leqslant \sum_{u \in V} g(u), \tag{13}$$

where $g(u) = \mathbb{E}$ (*the cost of grabbing all sons from* u *if* u *is the first vertex*).

The formal proof is left in Appendix B. Now it remains to estimate the right hand side of Equation (13). In Lemma 22, we have shown that if the graph is dense with all vertices having degree $\Omega(n)$, then $g(u) = O(\log n)$. Similar techniques apply to general graphs, and Theorem 8 is established by discussing the degree of each vertex and use different methods of counting based on it.

▶ **Theorem 8.** *For a graph* $G = (V, E)$ *with degrees of vertices sorted in ascending order* $d_1, d_2, \ldots, d_n$*, there is a linear-time algorithm that synthesizes a parity network for* G *with expected size*

$$m + O\left(\min_{1 \leqslant k \leqslant n} \left\{ \sum_{i \leqslant k} d_i + \frac{(n-k)n \log n}{d_{k+1}} \right\}\right). \tag{3}$$

Proof. Let us pick a threshold k, and use different methods of counting conditioning on the threshold. Then for each u,

1. If $\deg(u) \leqslant k$, then we assume all the neighbors will require a bridge, and add $\deg(u)$ to the overall cost.
2. If $\deg(u) > k$, then we consider all the neighbors v of u.
   a. If $\deg(v) \leqslant k$, then we assume v will require a bridge from u to its neighbor, and add 1 to the overall cost.
   b. If $\deg(v) > k$, then we can apply the same argument as in Lemma 22 that after $\Theta\left(\frac{n}{k} \log n\right)$ vertices, the probability that v is not grabbed is $O(n^{-1})$. Therefore, all these vertices will together contribute $\Theta\left(\frac{n}{k} \log n\right)$ to the overall cost.

Then summing up all the contributions,

- The overall contribution of 1. is $\sum_{d_i \leqslant k} d_i$.
- The overall contribution of 2.a is $\sum_{d_i \leqslant k} d_i$.
- The overall contribution of 2.b is $\Theta\left(\sum_{d_i > k} \frac{n}{k} \log n\right)$.

Therefore, (3) in Theorem 8 is in fact searching for the best threshold to minimize the overall estimated cost. ◀

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 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.

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

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

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

10 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.

11 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.

12 John Kallaugher, Ojas Parekh, and Nadezhda Voronova. Exponential Quantum Space Advantage for Approximating Maximum Directed Cut in the Streaming Model, November 2023. *To appear in STOC 2024*. arXiv:2311.14123.

13 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.

14 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.

15 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.

16 John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018. doi:10.22331/q-2018-08-06-79.

17 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.

18 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.

19 Robert E. Tarjan and Mihalis Yannakakis. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, August 1984. doi:10.1137/0213035.

20 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.

21 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.

22 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.

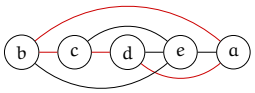
23 Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices. *Physical Review X*, 10(2):021067, June 2020. doi:10.1103/PhysRevX.10.021067.

A chordal $\subsetneq$ parity-traversable

By further investigating Algorithm 1, we develop the following condition for its success, which is weaker than G being chordal and π being a perfect elimination ordering of G.

▶ **Lemma 24.** *Algorithm 1 will synthesize an ideal parity network for* G *if* π *is an ordering that for each* i*, let* $\pi|_{\Gamma_i^+}$ *be* $\{\ell_1, \ell_2, \ldots, \ell_k\}$*, then there is a path* $\pi_{\ell_1} - \pi_{\ell_2} - \cdots - \pi_{\ell_k}$ *in* G.

An example of (G, π) that satisfies Lemma 24 is shown below with $\pi = (b, d, c, e, a)$, which is clearly not chordal (there is an induced 4-cycle $a - b - c - d$ colored red) but still admits an ideal parity network that is constructible using Algorithm 1.



We briefly illustrate the idea behind Lemma 24: consider π_i has two succeeding neighbors π_j and $\pi_k (i < j < k)$. Then when we process π_k, we will first generate $\{\pi_k, \pi_j\}$ on row π_j. Now we want to generate $\{\pi_k, \pi_i\}$, and we assert the current term of row π_i must be $\{\pi_j, \pi_i\}$ – because π_j is the last one that alters the term of row π_i since there are no neighbor of π_i between π_j and π_k, and therefore we can generate $\{\pi_k, \pi_i\}$ on row π_i.

Currently, it is not clear which graphs have such an ordering defined in Lemma 24. We leave it as a promising direction for future work. Moreover, we are also unaware whether all parity-traversable graphs have such an ordering, which is another interesting question to investigate.

B Proof of Lemma 23

Let $P(V, k) = \bigcup_{s \subset V, |s|=k} \mathrm{Sym}(s)$ be the set of all permutations of every k vertices in V, and $f(s \mid V \setminus s)$ where s is an ordering of a vertex subset, be the cost of grabbing all sons of s_1 when running the algorithm using order s conditioning on all vertices in $V \setminus s$ have been dealt with, then the overall cost is

$$\begin{aligned}
\mathbb{E}(K) =& \mathbb{E}_{\pi \in \mathrm{Sym}(n)} \left(f(\pi) + f\left(\pi|_{[2,n]}\right) + f\left(\pi|_{[3,n]}\right) + \ldots \right) \\
=& \sum_{k=1}^{n} \mathbb{E}_{s \in P(V,k)} \left(f(s \mid V \setminus s) \right) \\
\leqslant& \sum_{k=1}^{n} \mathbb{E}_{s \in P(V,n)} \left(f(s \mid V \setminus s) \right)
\end{aligned} \tag{14}$$

The last inequality is because if we append elements to a certain subsequence $s \in P(n, k)$, then $f(s \mid V \setminus s)$ will never decrease, as there will only be more neighbors of s_1, hence more bridges might need to be built. Finally,

$$\mathbb{E}(K) \leqslant n \cdot \mathbb{E}_{\pi \in \mathrm{Sym}(n)}(f(\pi)) = \sum_{u=1}^{n} g(u). \tag{15}$$