Cut Query Algorithms with Star Contraction

Simon Apers
CNRS, IRIF
Paris, France
smgapers@gmail.com

Yuval Efron
Columbia University
New York, USA
ye2210@columbia.edu

Pawel Gawrychowski
University of Wroclaw
Wroclaw, Poland
gawry@cs.uni.wroc.pl

Troy Lee
QSI, University of Technology Sydney
Sydney, Australia
troyjlee@gmail.com

Sagnik Mukhopadhyay
University of Sheffield
Sheffield, UK
s.mukhopadhyay@sheffield.ac.uk

Danupon Nanongkai
Max Planck Institute for Informatics,
University of Copenhagen, and KTH
danupon@gmail.com

Abstract—We study the complexity of determining the edge connectivity of a simple graph with cut queries. We show that (i) there is a bounded-error randomized algorithm that computes edge connectivity with $O(n)$ cut queries, and (ii) there is a bounded-error quantum algorithm that computes edge connectivity with $O(\sqrt{n})$ cut queries. To prove these results we introduce a new technique, called star contraction, to randomly contract edges of a graph while preserving non-trivial minimum cuts. In star contraction vertices randomly contract an edge incident on a small set of randomly chosen “center” vertices. In contrast to the related 2-out contraction technique of Ghaflari, Nowicki, and Thorup [SODA'20], star contraction only contracts vertex-disjoint star subgraphs, which allows it to be efficiently implemented via cut queries.

The $O(n)$ bound from item (i) was not known even for the simpler problem of connectivity, and it improves the $O(n \log^2 n)$ upper bound by Rubinstein, Schramm, and Weinberg [ITCS'18]. The bound is tight under the reasonable conjecture that the randomized communication complexity of connectivity is $\Omega(n \log n)$, an open question since the seminal work of Babai, Frankl, and Simon [FOCS'86]. The bound also excludes using edge connectivity on simple graphs to prove a superlinear randomized query lower bound for minimizing a symmetric submodular function. The quantum algorithm from item (ii) gives a nearly-quadratic separation with the randomized complexity, and addresses an open question of Lee, Santha, and Zhang [SODA'21]. The algorithm can alternatively be viewed as computing the edge connectivity of a simple graph with $O(\sqrt{n})$ matrix-vector multiplication queries to its adjacency matrix.

Finally, we demonstrate the use of star contraction outside of the cut query setting by designing a one-pass semi-streaming algorithm for computing edge connectivity in the complete vertex arrival setting. This contrasts with the edge arrival setting where two passes are required.

Index Terms—F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes, F.2 Analysis of Algorithms and Problem Complexity.

I. INTRODUCTION AND CONTRIBUTION

The minimization of a submodular function is a classic problem in combinatorial optimization. Over a universe $V$, a submodular function $f : 2^V \rightarrow \mathbb{R}$ is a function that satisfies $f(S) + f(T) \geq f(S \cap T) + f(S \cup T)$ for all subsets $S, T \subseteq V$. The submodular function minimization (SFM) problem is the task of computing $\min_{S \subseteq V} f(S)$. The SFM problem generalizes several well known combinatorial optimization problems such as computing the minimum weight of an $st$-cut in a directed graph and the matroid intersection problem. The SFM problem comes in another flavor when the submodular function $f$ is symmetric, i.e. also satisfies $f(S) = f(V \setminus S)$ for all $S$. In this case, $\emptyset$ and $V$ are trivial minimizers, so the interesting problem is to compute $\min_{S \subseteq V} f(S)$, the non-trivial minimum. The global minimum cut problem on an undirected graph is an instance of non-trivial symmetric submodular function minimization, which we denote by sym-SFM.

The size of the truth table of a submodular function is exponential in the size of $V$, so (sym-)SFM is typically studied in the setting where we have access to an evaluation oracle for $f$, that is, we can query any $S \subseteq V$ and receive the answer $f(S)$. When $|V| = n$, Grötschel, Lovász, and Schrijver [1] showed that the ellipsoid method can be used to solve SFM with $\tilde{O}(n^5)$ oracle queries and overall running time $\tilde{O}(n^7)$ [2, Theorem 2.8]. Since then a long line of work has developed faster and simpler (combinatorial) algorithms for SFM [3]–[8]. The work of [8] shows that SFM can be solved by a deterministic algorithm making $O(n^2 \log n)$ queries to an evaluation oracle. By the isolating cut lemma of [9] this immediately also gives an $O(n^3)$ query randomized algorithm for sym-SFM [10], [11]. In the deterministic case, the best upper bound on the number of queries to solve sym-SFM remains the $O(n^3)$ algorithm of Queyranne [12].

While sym-SFM is a much more general problem, it has a close relationship with one of its simplest instantiations: the global minimum cut problem. In this problem we are given a weighted and undirected graph $G = (V, E, w)$ and the task is to find the minimum weight of set of edges whose removal
disconnects $G$. For a subset $S \subseteq V$ let $\text{cut}_G(S)$ be the set of edges of $G$ with exactly one endpoint in $S$. The cut function $f : 2^V \to \mathbb{R}$, where $f(S) = w(\text{cut}_G(S))$ is the total weight of edges in $\text{cut}_G(S)$, is a symmetric submodular function. Evaluation queries in this case are called cut queries and the goal is to compute $\lambda(G) := \min_{\emptyset \subseteq S \subseteq V} w(\text{cut}_G(S))$ with as few cut queries as possible.

Both the best known deterministic and randomized algorithms for sym-SFM use ideas that originated in the study of minimum cuts: Queyranne’s algorithm is based on the Stoer-Wagner minimum cut algorithm [13], and the best randomized algorithm makes use of the isolating cut lemma originally developed in the context of a deterministic minimum cut algorithm [9]. On the lower bound side, the best known bounds on the query complexity of sym-SFM are $\Omega(n)$ in the deterministic case [14], [15] and $\Omega(n/\log n)$ in the randomized case [16] (see I). Both of these bounds can be shown for the cut query complexity of determining the weight of a minimum cut in a simple graph. The weight of a minimum cut in simple graph $G$ is known as the edge connectivity of $G$, and is the minimum number of edges whose removal disconnects the graph. The aforementioned lower bounds even hold for the more special case of determining if the edge connectivity is positive, i.e. if the graph is connected or not.

Recent work has given randomized algorithms that can compute $\lambda(G)$ with $O(n\log^3 n)$ cut queries in the case of simple graphs [17] and $n\log O(1)(n)$ cut queries in the case of weighted graphs [18]. For the deterministic case, however, the best upper bound remains $O(n^2/\log n)$ [19] and proceeds by learning the entire graph. Researchers continue to study the minimum cut problem as a candidate to show superlinear lower bounds on sym-SFM. Graur, Pollner, Ramaswamy and Weinberg [20] introduced a linear-algebraic lower bound technique known as the cut dimension to show a deterministic cut query lower bound of $3n^2/2 - 2$ for minimum cut on weighted graphs. Lee, Li, Santha, and Zhang [21] show that the cut dimension cannot show lower bounds larger than $2n - 3$, but use a generalization of the cut dimension to show a cut query lower bound of $2n - 2$, the current best lower bound known on sym-SFM in general.

Despite this work, showing a superlinear lower bound on the query complexity of sym-SFM remains elusive. In this paper, we show that for randomized algorithms and the special case of edge connectivity there is actually a linear upper bound.

**Theorem 1.1:** There is a randomized algorithm that makes $O(n)$ cut queries and outputs the edge connectivity of a simple input graph $G$ with probability at least $2/3$.

In particular one cannot hope to prove superlinear lower bounds on the randomized query complexity of sym-SFM via the edge connectivity problem. It remains open if Theorem 1.1 is tight. The best known lower bound is $\Omega(n \log \log(n)/\log(n))$ which follows from the $\Omega(n \log \log n)$ randomized communication complexity lower bound for edge connectivity by Assadi and Dudeja [22]. An $o(n)$ randomized cut query upper bound on edge connectivity would in particular imply a randomized communication complexity protocol for determining if a graph is connected with $o(n \log n)$ bits, resolving one of the longest standing open problems in communication complexity. Graph connectivity was a focus of many early works on communication complexity [14], [16], [23], and while a deterministic lower bound of $\Omega(n \log n)$ was established early on [14], to this day the randomized communication complexity is only known to be between $\Omega(n)$ and $O(n \log n)$.

Theorem 1.1 even improves the previous best cut query upper bound for deciding if a graph is connected. Harvey [15, Theorem 5.10] gave a deterministic $O(n \log n)$ cut query upper bound for connectivity, and we are not aware of any better upper bound in the randomized case. For the case of connectivity we can give a linear upper bound even for zero-error algorithms.

**Theorem 1.2:** Let $G = (V, E)$ be a simple $n$-vertex graph. There is a zero-error randomized algorithm that makes $O(n)$ cut queries in expectation and outputs a spanning forest of $G$.

The best lower bound we are aware of in this case is $\Omega(n \log \log(n)/\log(n))$ which follows from the non-deterministic communication complexity lower bound for connectivity of $\Omega(n \log \log(n))/\log(n))$ by Raz and Spiezker [23].

A key to both Theorem 1.1 and 1.2 is to think in terms of matrix-vector multiplication queries. If $A$ is the adjacency matrix of an $n$-vertex simple graph $G$, in a matrix-vector multiplication query we can query any vector $x \in \{0, 1\}^n$ and receive the answer $Ax$. If $G$ has maximum degree $d$, and so $A$ has at most $d$ ones in every row, we can learn the entire graph $G$ with only $O(d \log n)$ matrix-vector multiplication queries—this is one of the key ideas behind compressed sensing. As a single matrix-vector multiplication query can be simulated with $O(n)$ cut queries, this shows that we can learn $G$ with $O(n d \log n)$ cut queries. Grebinski and Kucherov [19] show the surprising fact that if $G$ is bipartite with maximum degree $d$, and the left and right hand sides are roughly the same size, then one can actually learn $G$ with only $O(\log d)$ cut queries. This savings of a $\log n$ factor over the trivial simulation is key to our improved algorithms.

We use this idea to design a primitive called Recover-k-From-All. Given two disjoint subsets $S, T$ of vertices, with the promise that all vertices in $S$ have at least $k$ neighbors in $T$, Recover-k-From-All makes $O(k n)$ cut queries and learns at least $k$ neighbors in $T$ of every vertex in $S$. This routine is the heart of our algorithm for Theorem 1.2, which uses it to implement Borůvka’s spanning forest algorithm.

It is less obvious how such a primitive is useful to compute edge connectivity as it only gives us local snapshots of sparse

1We say that a graph is simple if it is undirected and unweighted and contains at most one edge between any pair of vertices.

2With shared randomness the parties can simulate a randomized cut query algorithm with an $O(\log n)$ multiplicative overhead: whenever the algorithm makes a cut query, the parties communicate the number of cut edges in their part of the graph with $O(\log n)$ bits. By Newman’s theorem, this protocol can be simulated without shared randomness (and with only an additive $O(\log n)$ overhead).
star contraction

R will have a neighbor

R \delta V constant \in \text{ of size } \text{ vertices with high probability, where }

\text{ is the } \Theta(\text{ is the } \text{ will only have }

G = 3 \text{ \text{ and the edge connectivity of } V \text{ without too high a fraction of them being }

V \text{ if it isolates a single vertex.}

trivial without having too high a probability

G \in \text{ then the contracted }

k \text{ to choose a neighbor in } R \text{ BUT THE SAME ARGUMENT HOLDS FOR CUT QUERIES}

\Omega(\text{[19]} \text{\text{[19, S} \text{O} \text{ - that }} \text{ Lower }

\text{O} \text{[15]} \Omega(\text{[19]}) \text{ TO LEARN THE FULL GRAPH WITH }

\text{O} \text{[14]} \Omega(\text{[19]}). \text{ The fact that the edges that we want to }

contract are incident on a small number of vertices is a key

in independent sampling and choosing an edge incident to a

Recover- From-All because of the combination of requiring

in streamlining algorithms for graph problems [29], it began to

be studied in and of itself relatively recently in the work of

[30], and has since seen several several follow-ups [26],

[31]. More surprisingly, it turns out that the study of quantum

algorithms with cut query access to a graph is also closely

related to the matrix-vector multiplication model. This is

because with \text{O}(\log n) cut queries a quantum algorithm can simulate a restricted form of a matrix-vector multiplication

query, namely it can compute \text{Ax} in the entries where x is zero

(this is implicit in [32] and made explicit in [26, Corollary

11]). Lee, Santha and Zhang [32] used this to show that a

quantum algorithm making only \text{O}(\log^6 n) cut queries can decide if an n-vertex simple graph is connected [32, Theorem

44], a nearly exponential speedup over the best possible

randomized algorithm. They left it as an open question whether

any quantum speedup is possible for the problem of edge

connectivity. This problem is particularly interesting because

not much is known about the complexity of (sym)-SFM with

respect to quantum algorithms, either in terms of upper or

lower bounds (some work has been done on approximation

algorithms for SFM, see [33]). The best classical algorithms

for (sym)-SFM tend to be highly sequential, a feature which

is typically hard to speed up quantumly. Our matrix-vector

multiplication perspective leads to a quantum improvement

for the cut query complexity of edge connectivity, as shown

by the following theorem.

Theorem 1.3: There is a quantum algorithm that makes

\text{O}(\sqrt{n}) cut queries and outputs the edge connectivity of the input simple graph G correctly with high probability. Similarly, there is a randomized algorithm making \text{O}(\sqrt{n}) matrix-vector multiplication queries to the adjacency matrix of G that outputs the edge connectivity of the input simple graph G correctly with high probability.

The quantum part of this theorem gives a near-quadratic speedup over the best possible randomized algorithm. Moreover, there is a natural bottleneck to giving a \text{O}(\sqrt{n}) quantum cut query algorithm for edge connectivity, which is that even.

bipartite induced subgraphs. To this end we develop a new technique for edge connectivity called star contraction. Star contraction is inspired by the randomized 2-out contraction algorithm of Ghaffari, Nowicki and Thorup [27]. In that algorithm, each vertex independently and uniformly at random selects two incident edges. Ghaffari et al. show that when the selected edges are contracted the resulting graph \text{G’} has only \text{O}(n/\delta(G)) vertices with high probability, where \delta(G) is the minimum degree of \text{G}, and further with constant probability no edge of a non-trivial minimum cut is contracted. When these good things happen the edge connectivity of \text{G} is the minimum of \delta(G) and the edge connectivity of \text{G’}.

2-out contraction is not very compatible with our primitive Recover-k-From-All because of the combination of requiring independent sampling and choosing an edge incident to a vertex uniformly at random. Instead, in star contraction we first randomly choose a subset \text{R} of size \Theta(n \log(n)/\delta(G)). With high probability every vertex in \text{V} \setminus \text{R} will have a neighbor in \text{R}, and in star contraction we only contract edges with an endpoint in \text{R}. The fact that the edges that we want to contract are incident on a small number of vertices is a key to the savings of star contraction over 2-out contraction in the cut query model. Further, if for every vertex in \text{v} \in \text{V} \setminus \text{R} we contract an edge connecting it to \text{R} then the contracted graph \text{G’} will automatically have its size bounded by |\text{R}| = \Theta(n \log(n)/\delta(G)). While proving the contracted graph has few vertices is the most difficult part of the argument in 2-out contraction, for star contraction it is trivial (although the bound we get is larger by a \log n factor).

The tricky part remaining is how to choose a neighbor in \text{R} for each \text{v} \in \text{V} \setminus \text{R} without having too high a probability of choosing a neighbor on the other side of a non-trivial minimum cut. In our main technical contribution, we show that each \text{v} \in \text{V} \setminus \text{R} can learn just a constant number of neighbors in \text{R} without too high a fraction of them being on the opposite side of a non-trivial minimum cut. Moreover, we can allow correlations between the neighbors learned for different vertices which allows us to efficiently learn these constant number of neighbors using Recover-k-From-All with constant \text{k}.

\text{We call a cut trivial if it isolates a single vertex.}
computing the minimum degree of a graph seems to require \( \Omega(\sqrt{n}) \) quantum cut queries.\(^4\) There is a very natural \( O(\sqrt{n}) \) quantum algorithm for computing the minimum degree: the degree of a single vertex can be computed with one cut query, and one can then use quantum minimum finding [34] on top of this to find the minimum degree with \( O(\sqrt{n}) \) cut queries. We conjecture that this simple algorithm is optimal, which would imply that the quantum statement of Theorem 1.3 is tight up to polylogarithmic factors.

As a final application, we use our new star contraction technique to obtain a one-pass \( \tilde{O}(n) \)-space algorithm for computing edge connectivity with high probability in the complete vertex-arrival streaming model. In this model, the vertices of the graph \( G \) arrive in an arbitrary order with all incident edges. This contrasts with the edge-arrival streaming model, where edges of \( G \) arrive in arbitrary order, for which a \( \tilde{\Omega}(n^2) \) lower bound was proven on the space complexity of a one-pass algorithm that computes the edge connectivity [35]. This bound can be modified to also prove an \( \tilde{\Omega}(n^2) \) lower bound on the one-pass space complexity of edge connectivity in the more restrictive explicit vertex-arrival model, where the vertices of \( G \) arrive only with the edges incident on the previously seen vertices, as was considered in e.g. [36]. For completeness, we include a proof sketch of this lower bound in the appendix of the full version of the paper [24]. If however the vertices arrive with edges incident on the previously seen vertices in a random order, then our technique still implies an \( \tilde{O}(n) \)-space algorithm. For comparison, we also discuss why it is not clear how to use the related 2-out contraction technique to achieve these results.

II. TECHNICAL OVERVIEW

In the following sections we introduce one of the main tools in this work, star contraction, and give a sketch of the classical and quantum cut query algorithms that make use of star contraction.

A. Star contraction

The main workhorse for proving our results is a new technique for randomly contracting edges of a simple graph while preserving a non-trivial minimum cut with constant probability. The idea of contracting edges while preserving non-trivial minimum cuts comes from the celebrated result of Kawarabayashi and Thorup (Fulkerson Prize 2021) [37], which gave the first near-linear time deterministic algorithm for computing the edge connectivity of a simple graph. A critical observation in their work is the following: we can contract edges in a simple graph \( G \) to get a graph \( G' \) so that (i) \( G' \) has \( \tilde{O}(n/\delta(G)) \) vertices and \( \tilde{O}(n) \) edges, where \( \delta(G) \) is the minimum degree of \( G \), and (ii) all non-trivial minimum cuts in \( G \) are preserved (i.e., no edge participating in a non-trivial minimum cut is contracted). In particular, if \( G \) has a non-trivial minimum cut then \( \lambda(G') = \lambda(G) \).

Such a contraction is useful for computing edge connectivity since when the edge connectivity is large and there is a non-trivial minimum cut (which is usually harder to handle), the contraction significantly sparsifies and reduces the number of vertices of the graph. We call this type of contraction a KT contraction.

The KT contraction technique has been highly influential, and many works have since used and studied it. The algorithm for KT contraction given in [37] takes time \( O(m \log^2 n) \) in the sequential setting when the graph has \( m \) edges. This was later improved by Henzinger, Rao, and Wang [38] to \( O(m \log^2 n (\log \log n)^2) \). Using an expander decomposition algorithm [39]–[42] as a black box, Saranurak [43] showed a slower but simpler \( \tilde{O}(m^{1+\omega(1)}) \) time algorithm to compute a KT contraction. All these algorithms are deterministic but rather complicated, making them hard to adapt to other settings. Rubinstein, Schramm, and Weinberg [17] provide a randomized algorithm for computing a KT contraction that is efficient in the cut-query setting, and leads to their aforementioned \( O(n \log^3 n) \) cut query algorithm for edge connectivity.

Most relevant for our work is the beautiful 2-out contraction algorithm by Ghaffari, Nowicki, and Thorup [27]. In this algorithm, every vertex independently at random (with replacement) chooses two of its incident edges to contract. Ghaffari et al. show that the resulting contracted graph \( G' \) has only \( O(n/\delta(G)) \) vertices with high probability, and moreover if \( G \) has a non-trivial minimum cut then \( \lambda(G) = \lambda(G') \) with constant probability. They use this algorithm to get the current fastest randomized algorithm for edge connectivity with runtime\(^5\) \( O(\min\{m + n \log^2 n, m \log n\}) \), and they also obtain improved algorithms for edge connectivity in the distributed setting.

Although [27] did not study the cut query model, the 2-out contraction approach gives a simple randomized algorithm for edge connectivity with \( O(n \log n) \) cut queries, improving the bound from [17]. As this is very related to our approach, we give an outline of the proof here. First, we can compute \( \delta(G) \) with \( n \) cut queries by querying \( \lceil \text{cut}(|v|) \rceil \) for every vertex \( v \). The next thing to notice is that for any vertex \( v \) we can randomly choose a neighbor of \( v \) with \( O(\log n) \) cut queries using a randomized version of binary search. This is because with 3 cut queries we can compute \( |E(v, S)| \) for any set \( S \subseteq V \setminus \{v\} \), and thus can continue searching for a neighbor of \( v \) in the set \( S \) with probability proportional to this number. Thus with \( O(n \log n) \) queries we can perform 2-out contraction and identify the sets of vertices forming the “supervertices” of the contracted graph \( G' \). By the main theorem of [27], with high probability \( G' \) will have \( O(n/\delta(G)) \) supervertices. The remaining task is to compute the edge connectivity of \( G' \). To do this we can make use of a very useful tool developed by Nagamochi and Ibaraki [45] called a sparse r-edge connectivity certificate. Let \( F \) be the set of edges found by repeating \( r \) times: (i) find a spanning forest

\(^4\)It is intuitive that computing the edge connectivity of a graph is more difficult than computing the minimum degree, and we formalize this via a simple reduction in the appendix of the full version of the paper [24].

\(^5\)The stated bound in [27] is \( O(\min\{m + n \log^3 n, m \log n\}) \), but more recent work on the minimum cut problem by [44] improves it to the bound we quote here.
of \( G' \) and (ii) add the edges of this spanning forest to \( F \) and remove them from \( G' \). Then Nagamochi and Ibaraki show that if \( |\text{cut}_G(S)| \leq r \) then \( |\text{cut}_G(S)| = |\text{cut}_F(S)| \). In particular, the edge connectivity of a sparse \( \delta(G) \)-edge connectivity certificate of \( G' \) will equal \( \tau = \min\{\delta(G), \lambda(G')\} \). Contraction cannot decrease edge connectivity, so if \( \lambda(G) = \delta(G) \) then \( \tau \) will always be the correct answer; if \( \lambda(G) < \delta(G) \) then it will be correct whenever we do not contract an edge of a non-trivial minimum cut in the 2-out contraction, which happens with constant probability. We can find a single spanning forest of \( G' \) deterministically with \( O(n \log(n)/\delta(G)) \) cut queries [15, Theorem 5.10], thus we can find a sparse \( \delta(G) \)-edge connectivity certificate of \( G' \) with \( O(n \log n) \) cut queries overall.

It is not obvious how to independently sample a uniformly random neighbor of every vertex without spending \( \Omega(\log n) \) cut queries on average per vertex. Even with the very powerful matrix-vector multiplication queries, where one can learn the entire neighborhood of a vertex with a single query, it is not clear what else one can do to avoid spending \( \Omega(1) \) queries per vertex on average to implement 2-out contraction.

We introduce a new graph contraction technique called star contraction that allows one to take advantage of the power of cut and matrix-vector multiplication queries to process vertices in parallel. For greater flexibility in the applications to different types of queries, we state this as a general method that can be instantiated in various ways.

**Technique 2.1 (Star contraction):** Let \( G = (V, E) \) be a simple graph and \( p \in (0, 1] \) be a probability parameter (think of \( p \in \Theta(1/\delta(G)) \)).

1. Define a set of “center vertices” \( R \) where every vertex is put into \( R \) independently at random with probability \( p \).

2. Define a set of “star edges” \( X \) by doing the following for every vertex \( v \notin R \): pick a neighbor \( c \in R \) (if it exists) and put the edge \( \{v, c\} \) into \( X \). The set \( X \) is a collection of star subgraphs centered at vertices in \( R \).

Output the graph \( G' \) obtained from \( G \) by contracting all edges in \( X \).

Note that in item 2 we do not specify how to pick a neighbor in \( R \). The rule for doing this will vary in our applications. The nice thing about the star contraction framework is that no matter what rule is used here, the number of vertices in \( G' \) will always be at most \( |R| \) plus the number of vertices in \( V \setminus R \) that have no neighbor in \( R \). By taking \( p = \Theta(\log(n)/\delta(G)) \), for example, with high probability all vertices will have a neighbor in \( R \) and hence \( G' \) will only have \( O(n \log(n)/\delta(G)) \) many vertices. This leaves one only with the question of choosing a good rule to instantiate item 2 that does not choose an edge of non-trivial minimum cut with too high a probability, and that can be efficiently performed in the query model of interest.

The most natural rule to instantiate item 2 of Technique 2.1 is to have each vertex in \( V \setminus R \) independently and uniformly at random choose a neighbor in \( R \). We refer to this instantiation as uniform star contraction. The analysis of this case suffices for our algorithms in the quantum cut query model, the matrix-vector multiplication query model, and the streaming model. As an example, in the matrix-vector multiplication and quantum cut query settings we can learn all the neighbors of a vertex with \( \tilde{O}(1) \) queries. Hence, we can learn all edges incident on the center vertices \( R \) with only \( \tilde{O}(|R|) \) queries, which then allows us to implement uniform star contraction. Now if \( \delta(G) \) is large (and we choose \( p = \Theta(\log(n)/\delta(G)) \) as above) then the query cost \( \tilde{O}(|R|) \) will be small, and we will take advantage of this. This contrasts with the case of 2-out contraction, where in general there does not exist a small set so that all contracted edges are incident on this set (as an example, consider the case where \( G \) is the complete graph).

Formally, we show the following theorem about uniform star contraction.

**Theorem 2.2:** Let \( G = (V, E) \) be an \( n \)-vertex simple graph with \( \lambda(G) < \delta(G) \). Then uniform star contraction with \( p = \frac{1200 \ln n}{\delta(G)} \) gives \( G' \) where

1. \( G' \) has at most \( 2400 \ln n / \delta(G) \) vertices with probability at least \( 1 - 1/n^4 \), and

2. \( \lambda(G') = \lambda(G) \) with probability at least \( 2 \cdot 3^{-13} \).

We give an overview of the proof here. As mentioned, the number of vertices in the contracted graph \( G' \) is at most \( |R| \) plus the number of vertices in \( V \setminus R \) that have no neighbor in \( R \). By a Chernoff bound, with high probability \( |R| \) will be at most twice its expectation, which is \( \Theta(n \log(n)/\delta(G)) \). Further, the expected number of neighbors of any vertex \( v \) among the centers \( R \) is \( \Omega(\log n) \). Thus, by a Chernoff bound plus a union bound, every vertex in \( V \setminus R \) will have a neighbor in \( R \) with high probability. This argument nearly trivially bounds the number of vertices in \( G' \). In contrast, bounding the number of vertices in \( G' \) is the most complicated part of the proof for the analogous statement for 2-out contraction, although it must be noted the bound obtained there is better by a factor of \( \log n \). Another nice property of star contraction is that each connected component of \( G' \) has diameter 2, a property that is useful for designing algorithms in models of distributed computing. Ghaflari et al. show that after 2-out contraction the average diameter of a component is \( O(\log \delta(G)) \) with high probability [27, Lemma 5.1].

For the second item of the theorem it is useful to first review the proof of the analogous statement for 2-out contraction. Let \( C \) be a non-trivial minimum cut of \( G \). Let a random \( 1 \)-out sample of \( G \) be the set of edges obtained by independently and uniformly at random selecting an edge incident to each vertex. A 2-out contraction is exactly the process of independently performing two random 1-out samples of \( G \) and contracting all the selected edges. The probability that we contract an edge of \( C \) in performing a 2-out contraction is exactly the square of the probability that we select an edge of \( C \) in a random 1-out sample.

For every vertex \( v \) let \( d(v) \) be the degree of \( v \) and \( c(v) \) be the number of \( u \) such that \( \{u, v\} \in C \). Let \( N(C) = \{v \in V : \}

\[\text{Perhaps more comparable to our case, Ghaflari et al. also obtain a worst-case upper bound of} \ O(\log n) \ \text{on the diameters of the connected components of a contracted graph} \ G' \ \text{with} \ O(n \log(n)/\delta(G)) \ \text{vertices obtained by only contracting a subset of the edges selected in a 2-out sample} [27, Remark 5.3].\]
$c(v) > 0 \}$ be the set of vertices incident to $C$. When we take a random 1-out sample of $G$, the probability that we do not choose an edge of $C$ is exactly

$$\prod_{v \in N(C)} \left(1 - \frac{c(v)}{d(v)}\right). \tag{1}$$

At first it might seem that this probability could be very small. The key to lower bounding it combines two observations:

$$\frac{c(v)}{d(v)} \leq 1/2 \text{ for every } v \in N(C) \tag{2}$$

$$\sum_{v \in N(C)} \frac{c(v)}{d(v)} \leq \frac{|C|}{\delta(G)} \leq 2. \tag{3}$$

The first inequality follows from the fact that $C$ is a non-trivial cut, and if (2) did not hold then we could move $v$ to the other side and obtain a smaller cut. To derive (3) we use the fact that $d(v) \geq \delta(G)$ and $|C| \leq \delta(G)$.

How small can (1) be under the constraints of (2) and (3)? In fact it is always at least $1/16$; some thought shows that the minimum of (1) will be achieved at an extreme point of the set of constraints, which is obtained when 4 vertices have $c(v)/d(v) = 1/2$, thereby saturating both (2) and (3). The lower bound of $1/16$ is tight as can be seen by taking a non-trivial minimum cut of the cycle graph. This completes the slick argument that indeed 2-out contraction will not contract an edge of a non-trivial minimum cut with constant probability.

Correctness of our algorithms based on star contraction is proven using analogs of (2) and (3) (with slightly worse constants). We again illustrate the correctness proof for the case of uniform star contraction. Let $d_R(v) = |E(v, R \setminus \{v\})|$ and $c_R(v) = |C \cap E(v, R \setminus \{v\})|$. As an analog of (3), we directly show that for all $v \in V$

$$\mathbb{E}_R \left[ \frac{c_R(v)}{d_R(v)} \mid d_R(v) > 0 \right] = \frac{c(v)}{d(v)}. \tag{4}$$

Therefore by linearity of expectation and 3 we have $\mathbb{E} \left[ \sum_{v \in N(v)} c_R(v)/d_R(v) \right] \leq 2$, and by Markov’s inequality the sum will not significantly exceed this quantity with constant probability. To prove an analog of (2) we again use the fact that, with high probability, $d_R(v) = \Omega(\log n)$ for all $v$. By (4) we also know that $\mathbb{E}_R[c_R(v)/d_R(v) \mid d_R(v) > 0] = c(v)/d(v) \leq 1/2$. Thus for $c_R(v)/d_R(v)$ to exceed $2/3$ we must have $c_R(v) = \Omega(\log n)$ and $c_R(v)$ exceeding its expected value by a constant factor bigger than 1. We then again use a Chernoff bound to show that for each $v$ individually this does not happen with high probability, and finally apply a union bound over all $v$.

**B. Matrix-vector multiplication and quantum cut query algorithm**

As a direct application of our uniform star contraction procedure we obtain an algorithm for computing the edge connectivity of a simple graph with $\tilde{O}(\sqrt{n})$ quantum cut queries (i.e., Theorem 1.3) or matrix-vector multiplication queries to the adjacency matrix. We sketch the algorithm here and refer the reader to the full version of the paper [24] for a complete description and analysis of the algorithm.

The algorithm uses the following primitives (which we can run either on the original graph, a vertex-induced subgraph, or a vertex-induced subgraph with an explicit set of edges removed):

**P1. Find all neighbors of a vertex.** This can be done with 1 matrix-vector multiplication query to the adjacency matrix (for a vertex $v$, query $A\chi_v$ with $\chi_v$ the standard basis vector corresponding to vertex $v$) or $O(\log n)$ quantum cut queries (this is implicitly shown in [32] and made explicit in [26, Corollary 11]).

**P2. Construct a spanning forest.** This can be done with polylog$(n)$ matrix-vector multiplication queries [26] or polylog$(n)$ quantum cut queries [32].

**P3. Compute the minimum degree.** This takes 1 matrix-vector multiplication query (query for the matrix-vector product $A1$, with 1 the all-ones vector) or $O(\sqrt{n})$ quantum cut queries (run quantum minimum finding on the vertex degrees).

**P4. Compute a cut query.** This can clearly be done with 1 matrix-vector multiplication or quantum cut query.

Uniform star contraction for a given parameter $p$ can be implemented with just the first primitive: (i) pick a random subset of vertices $R$ by including every vertex independently at random with probability $p$, (ii) for every vertex in $R$ learn all its neighbors, explicitly giving the set $\text{cut}(R)$, and (iii) for every vertex $v$ not in $R$, select a random edge in $\text{cut}(R)$ incident to $v$ (if it exists). Contracting the resulting star graphs yields the (supervertices of) the contracted graph $G'$. We only make queries in step (ii). By primitive P1, this can be done with $O(\lceil|R|\rceil)$ queries, and this is $O(np)$ in expectation.

We can now easily sketch our algorithm for computing the edge connectivity of the input graph $G$ using the above primitives:

1) Compute the minimum degree $\delta(G)$ using primitive P3.

2) If $\delta(G) \leq \sqrt{n}$, find a sparse $\delta(G)$-edge connectivity certificate using primitive P2. Output the edge connectivity of the connectivity certificate.

3) If $\delta(G) > \sqrt{n}$, do uniform star contraction with $p = \Theta(\log(n)/\delta(G))$, resulting in a contracted multigraph $G'$ that has $\tilde{O}(\sqrt{n})$ vertices with high probability. Run the randomized algorithm from [18] (requiring $O(|V'|)$ cut queries) to compute $\lambda(G')$.

Step 1 can be implemented with $O(\sqrt{n})$ matrix-vector multiplication or quantum cut queries by P3. Step 2. costs polylog$(n)$ queries per spanning forest by P2, thus $O(\delta(G)) \in O(\sqrt{n})$ queries overall. In step 3. we have $|R| = O(\sqrt{n} \log n)$ with high probability, in which case the star contraction can be done with $O(\sqrt{n} \log n)$ queries by P1. The algorithm of [18] can compute the weight of a minimum cut in a weighted $N$-vertex graph with high probability after $\tilde{O}(N)$ classical cut

---

1 See the full version [24] for a proof. One can alternatively obtain a looser bound by using the inequality $1 - x \geq \exp(-x/(1-x))$ for $0 < x < 1$, as is done in [27].
queries, thus by primitive P4 we can compute \( \lambda(G') \) with \( \tilde{O}(\sqrt{n}) \) queries.

C. Randomized cut-query algorithm

Finally we describe our randomized \( O(n) \) cut query algorithm for edge connectivity. It does not seem possible to achieve this result using uniform star contraction as we did in the quantum cut query and matrix-vector multiplication query case. The reason is that a vertex in \( V \setminus R \) could have up to \( |R| \) many neighbors in \( R \) and it is too expensive to learn all of these neighbors with cut queries. Instead we use another variation on star contraction that we call sparse star contraction. We show that instead of choosing a uniformly random neighbor in \( R \), we can instead first learn a bipartite subgraph between \( V \setminus R \) and \( R \) where each vertex in \( V \setminus R \) has constant degree. We then do 1-out contraction by independently choosing, for each \( v \in V \setminus R \), a uniformly random neighbor in this sparse bipartite subgraph. Our main technical contribution is to show that this process can be done while preserving a non-trivial minimum cut with constant probability. We call this technique sparse star contraction as the contraction is performed on a bipartite subgraph with only \( O(n) \) edges. To actually learn this sparse bipartite subgraph with \( O(n) \) cut queries we use our second main tool, which is the separating matrix framework of Grebinski and Kucherov [19], [46]. We next elaborate on sparse star contraction and the separating matrix framework in more detail.

a) Sparse star contraction: To put sparse star contraction into context we begin with a more general scenario. We can imagine a general form of a randomized contraction algorithm that first learns a spanning subgraph \( \tilde{H} = (V_E') \) of the input graph \( G = (V,E) \), and then for each vertex \( v \in V \) independently at random selects an edge incident to \( v \) in \( H \). Finally, the selected edges are contracted in the original graph \( G \). In our case it will further be useful to think of \( H \) as a directed graph, where we will choose a random outgoing edge from each vertex. This point of view gives us more control over which endpoints can contract an edge. We make the following definition.

Definition 2.3 (Directed subgraph, 1-out contraction): Let \( G = (V,E) \) be a simple graph. We say that the directed graph \( H = (V,A) \) is a directed subgraph of \( G \) if every arc \( (u,v) \in A \) satisfies \( \{u,v\} \in E \). In a random 1-out sample of \( H \), we independently and uniformly at random choose an outgoing edge in \( H \) from every vertex that has one. In a random 1-out contraction of \( H \), we take a random 1-out sample of \( H \) and output the graph \( G' \) obtained by contracting the sampled edges in \( G \).

Note that in a directed subgraph \( H \), for any edge \( \{u,v\} \) of \( G \), we can either have both arcs \( (u,v),(v,u) \) in \( H \), just one of them, or neither of them. When we speak about doing 1-out contraction on an undirected subgraph it should be interpreted that all edges are oriented in both directions.

With this terminology, uniform star contraction corresponds to doing a random 1-out contraction on the directed subgraph \( H \) which is the induced bipartite graph between \( V \setminus R \) and \( R \) with all edges directed from \( V \setminus R \) to \( R \).

There are two properties that we want in a directed subgraph \( H \). The first is that after a random 1-out contraction on \( H \) the contracted graph does not have too many vertices. We can automatically guarantee this condition by working in the star contraction framework. The second is that in taking a random 1-out sample of \( H \) we do not have too high probability of selecting an edge of a non-trivial minimum cut. We can precisely extract a sufficient condition that makes a directed subgraph \( H \) “good for contracting” in this second sense.

Definition 2.4 ((\( \alpha, \beta \))-good for contracting): Let \( G = (V,E) \) be a simple graph and \( C \subseteq E \). Let \( H = (V,A) \) be a directed subgraph of \( G \). For every \( u \in V \) let \( q_u = \Pr_{v,(u,v) \in A} [\{u,v\} \in C] \) if \( |\{(u,v) \in A : v \in V\}| > 0 \) and \( q_u = 0 \) otherwise. We say that \( H \) is \( (\alpha, \beta) \)-good for contracting with respect to \( C \) if it satisfies the following two conditions

1) max property: \( \max_u q_u \leq \alpha \), and
2) sum property: \( \sum_u q_u \leq \beta \).

An undirected subgraph of \( G \) is \( (\alpha, \beta) \)-good for contracting if and only if its directed version where all edges are directed in both directions is.

As an example, it follows from (2) and (3) used in the correctness proof of 2-out contraction that \( G \) itself is \((1/2, 2)\)-good for contracting for any non-trivial minimum cut \( C \). In the full version of the paper [24], we show that if \( H \) is \((\alpha, \beta)\)-good for contracting with respect to \( C \) then the probability we do not select an edge of \( C \) in a random 1-out sample of \( H \) is at least \( (1 - \alpha)^{\beta/\alpha} \).

In sparse star contraction, we again start out by choosing a random set \( R \) by taking each vertex \( v \) to be in \( R \) with probability \( p \), although we take \( p = \Theta(\log(\delta(G))/\delta(G)) \) to be smaller than what we used before. With constant probability the number of vertices in \( V \setminus R \) with no neighbor in \( R \) will be \( O(n/\delta(G)) \), and \( R \) itself will satisfy \( |R| = O(n \log(\delta(G))/\delta(G)) \). Let \( H \) be the induced bipartite subgraph between \( V \setminus R \) and \( R \) with all edges directed from \( V \setminus R \). In uniform start contraction we do a random 1-out contraction on \( H \). For the randomized cut query algorithm we will learn a sparse subgraph \( H' \) of \( H \) that has the property that every \( v \in V \setminus R \) that has an outgoing edge in \( H \) also has an outgoing edge in \( H' \). No matter what \( H' \) we take with this property we are guaranteed that after 1-out contraction the resulting contracted graph will have \( O(n \log(\delta(G))/\delta(G)) \) vertices. Our main technical contribution (See full version for formal statement [24]) shows that we can find such an \( H' \) that is \((\alpha, \beta)\)-good for contracting for \( \alpha < 1 \) and small constant \( \beta \) that has constant degree. As \( H' \) only has \( O(n) \) edges, we can hope to learn it with \( O(n) \) cut queries, and we show this can indeed be done using the separating matrix machinery, described next.

b) Separating matrices and Recover-k-From-All: The second key tool of our algorithm is the separating matrix machinery. This toolset is best described by first considering an immediate obstacle to our \( O(n) \) cut query algorithm for edge
connectivity: an $O(n)$ bound is not even known for the simpler problem of determining if a graph is connected. Harvey gave a deterministic $O(n \log n)$ cut query algorithm for connectivity [15, Theorem 5.10], and we are not aware of any better result in the randomized case. Besides the fact that connectivity is a special case of edge connectivity, our algorithmic framework will also heavily rely on being able to efficiently find spanning forests to construct sparse $r$-edge connectivity certificates.

Harvey’s connectivity algorithm, which can also find a spanning forest, is an implementation of Prim’s spanning forest algorithm in the cut query model. This algorithm can equally well be implemented with a weaker oracle that simply reports whether or not there exists an edge between two disjoint sets $S$ and $T$ (this is known in the literature as a bipartite independent set oracle, see e.g. [47]). Interestingly, Harvey’s algorithm is actually optimal if restricted to this type of queries. Indeed, any deterministic algorithm that determines connectivity while making use of an oracle that returns 1 bit of information must make $\Omega(n \log n)$ queries. This follows from the aforementioned deterministic $\Omega(n \log n)$ 2-party communication complexity lower bound for deciding if a graph is connected [14].

The key now to both our zero-error $O(n)$ cut query algorithm for finding a spanning forest and our randomized $O(n)$ cut query algorithm for edge connectivity is to make use of the fact that a cut query actually returns $\Omega(\log n)$ bits of information. This power was first harnessed by Grebinski and Kucherov [19], [46] who studied a related, but more powerful, query known as an additive query. In this model, when the input is a simple $n$-vertex graph with adjacency matrix $A$ one can query two Boolean vectors $x, y \in \{0, 1\}^n$ and receive the answer $x^tAy$. Grebinski and Kucherov [19] showed the surprising fact that one can learn an $n$-vertex simple graph with only $O(n^2/\log n)$ additive queries, achieving the information theoretic lower bound. The main tool in the proof of Grebinski and Kucherov is the use of separating matrices: the existence of an $O(n/\log n)$-by-$n$ matrix $B$ such that $Bx \neq By$ for any two distinct $n$-dimensional Boolean vectors $x$ and $y$. We use the separating matrix framework of Grebinski and Kucherov to show that if $S,T$ are disjoint subsets of $V$ that are polynomially related in size and $d_T(v) \leq \ell$ for every $v \in S$, then we can learn all edges between $S$ and $T$ with only $O(\ell|S|)$ cut queries. See the full version for further details [24].

This fact is the heart of the subroutine Recover-$k$-From-All (See full version [24]) which plays a key role in both the spanning forest and edge connectivity algorithms. The input to this algorithm is two disjoint subsets $S,T \subseteq V$ that are polynomially related in size with the promise that $d_T(v) \geq \ell$ for every $v \in S$. Recover-$k$-From-All is a zero-error randomized algorithm that can then learn $\min\{k, \ell\}$ neighbors in $T$ for every vertex $v \in S$ and makes $O(k|S|)$ cut queries in expectation. Recover-$k$-From-All is based on ideas from $t_0$ sampling (e.g. Theorem 2 of [49]), which is similarly used in the connectivity algorithm in the semi-streaming model by Ahn, Guha, and McGregor [29]. First we put vertices in $S$ into $O(\log n)$ buckets by putting together those vertices with similar values of $d_T(v)$. For the bucket $B$ with degree around $r$ into $T$, we randomly subsample a set $T' \subseteq T$ by putting each vertex of $T$ into $T'$ with probability $2k/r$. We call a vertex in the bucket “caught” if $d_T(v)$ is close to its expectation (e.g., it is in $[k, 8k]$). Letting $B' \subseteq B$ be the set of caught vertices, we can then learn $E(B', T')$ with $O(k|B'|)$ cut queries. This is repeated on all buckets until all vertices have been caught. As we expect to catch a constant fraction of the remaining vertices in a bucket with each iteration, and the complexity of an iteration scales with the number of remaining vertices, one can argue that the expected number of queries overall is $O(kn)$.

c) Edge connectivity: Now that we described the main tools, we can describe the main algorithm. To make the exposition simpler, we begin with explaining how star contraction and Recover-$k$-From-All can be used to give a randomized $O(n \log \log n)$ cut query algorithm for edge connectivity. We then describe the additional trick needed to reduce the query complexity to $O(n)$. Both of these algorithms and their analysis can be found in the full version of the paper [24]. The basic algorithm essentially follows the same steps as used in the quantum cut query case.

1) Compute the minimum degree $\delta(G)$.
2) Perform sparse star contraction:

   a) Choose a set $R$ by taking each $v \in V$ to be in $R$ with probability $\Theta(\log(\delta(G))/\delta(G))$. Let $H$ be the directed subgraph obtained by picking every edge between $V \setminus R$ and $R$ and directing it from $V \setminus R$ to $R$.
   b) Use Recover-$k$-From-All with constant $k$ on $V \setminus R$ and $R$ to learn a sparse subgraph $H'$ of $H$.
   c) Do a random 1-out contraction on $H'$ and let $G'$ be the resulting graph.
3) Compute the edge connectivity of $G'$, and output the minimum of this and $\delta(G)$.

As in the proof of Theorem 2.2, we can again argue that with constant probability (i) $H$ is $(\alpha, \beta)$-good for contracting for some $\alpha < 1$ and constant $\beta$ (more specifically, for $\alpha = 3/5$ and $\beta = 8$) with respect to a non-trivial minimum cut, and (ii) that only $O(n/\delta(G))$ vertices in $V \setminus R$ have no neighbor in $R$. Now, however, it is too expensive to learn the entire subgraph $H$ as in the quantum cut query algorithm, or even to independently sample a uniformly random neighbor in $R$ for each $v \in V \setminus R$ within the $O(n)$ query budget. Instead, we use Recover-$k$-From-All with constant $k$ to learn a sparse subgraph $H'$ of $H$, where $H'$ has an outgoing edge for every $v \in V \setminus R$.

8We assume that $B', T'$ are polynomially related in size for this high level description. Handling smaller $B'$ is a technicality postponed to the full proof.
that has one in \( H \). Moreover, the outgoing neighbors of \( v \) in \( H' \) are learned from a random set of vertices, conditioned on this set having at least one and not too many neighbors of \( v \). This can be done with \( O(n) \) queries. We then do a random 1-out contraction on the explicitly known graph \( H' \). Letting \( G' \) be the result of this contraction, we finally compute the edge connectivity of \( G' \) and output the minimum of this and \( \delta(G) \).

We postpone describing how we compute the edge connectivity of \( G' \) and instead focus on showing that \( H' \) is still \((\alpha', \beta')\)-good for contracting for some \( \alpha' < 1 \) and constant \( \beta' \). As we use a constant \( k \) in Recover-\( k \)-From-All, we only expect to find a constant number of neighbors of a particular vertex \( v \). We have to show that, even in this very small sample, not too high a fraction of neighbors are on the opposite side of a non-trivial cut from \( v \), for all vertices \( v \) incident on the cut. In this low probability sampling regime, a Chernoff bound can only upper bound the failure probability for a particular vertex by a constant, which is not good enough as we have to union bound over the possibly growing number of vertices incident on the cut.

Instead, in the full version [24], we show the following statement. Let \( v \in V \setminus R \) and \( C \) be a non-trivial minimum cut of \( G \). Let \( R' \subseteq R \) be chosen by putting each vertex of \( R \) into \( R' \) with probability \( p = 2k/d_R(v) \) conditioned on \( d_R(v) > 0 \). We have already mentioned the fact that \( E_{R'}[c_{R'}(v)/d_{R'}(v) \mid d_{R'}(v) > 0] = c_R(v)/d_R(v) \). We show that as long as \( k \geq 10 \)

\[
\Pr_{R'}\left[\frac{c_{R'}(v)}{d_{R'}(v)} \geq \frac{c_R(v)}{d_R(v)} + \frac{1}{10} \mid d_{R'}(v) > 0 \right] \leq \frac{200 c_{R'}(v)}{k d_{R}(v)}.
\]

We know that \( \sum_{v \in R}(v) d_{R}(v) \leq 8 \) since \( H \) is \((3/5, 8)\)-good for contracting. Hence, by relating the failure probability to a sum that is bounded, and taking \( k \) to be a large constant, we can again use a union bound to argue that \( H' \) satisfies the max property with \( \alpha' = 7/10 \) with constant probability.

After a random 1-out contraction on \( H' \) the contracted graph \( G' \) has \( O(n \frac{\log(\delta(G))}{\delta(G)}) \) vertices. Ideally, however, we would like it to only have \( O(n/\delta(G)) \) vertices. As we describe next, we can then compute \( \lambda(G') \) with \( O(n) \) queries by finding a sparse \( \delta(G) \)-edge connectivity certificate. To further reduce the size of the contracted graph, we use Recover-\( k \)-From-All with \( k = \Theta(\log(\delta(G))) \) to learn a directed subgraph \( H_2 \) of \( G[R] \) where all but \( O(n/\delta(G)) \) vertices have outdegree \( h = \Omega(\log(\delta(G))) \). This requires \( O(\log(\delta(G)) \| R \|) = O(n) \) cut queries. As was done with \( H' \) we can similarly argue that \( H_2 \) is \((\alpha, \beta)\)-good for contracting for yet other \( \alpha < 1 \) and constant \( \beta \). We then do a 2-out contraction on \( H_2 \). We can use a lemma of [27, Lemma 2.5] to conclude that 2-out contraction on \( H_2 \) reduces the number of vertices in \( G[R] \) by a factor of \( h \) and thus becomes \( O(n/\delta(G)) \). As all but \( O(n/\delta(G)) \) vertices in \( V \setminus R \) are connected to a vertex in \( R \) this reduces the number of vertices in \( G' \) overall to \( O(n/\delta(G)) \).

d) \textbf{Spanning forests and sparse edge connectivity certificates.} In order to accomplish step 3. of the algorithm we show that we can construct a sparse \( r \)-edge connectivity certificate in a contracted graph with \( q \) vertices using \( O(n + rq \log(n)/\log(q)) \) cut queries. This lets us construct a sparse \( \delta(G) \)-edge connectivity certificate of \( G' \) with \( O(n) \) queries when \( G' \) has \( O(n/\delta(G)) \) vertices. In the final part of this section we give an overview of the key ideas that go into this algorithm and the obvious prerequisite of constructing a spanning forest with \( O(n) \) cut queries.

Our spanning forest algorithm follows the framework of Borůvka’s spanning forest algorithm as has been used in several works related to matrix-vector multiplication queries [26, 29, 32]. The application here requires several additional tricks to stay within the \( O(n) \) query budget.

The algorithm proceeds in rounds and maintains the invariant that in each round there is a partition \( S_1, \ldots, S_i \) of \( V \) and a spanning tree for each \( S_i \) in the partition. Initially, each \( S_i \) is just a single vertex. In each round, it performs the following two steps:

1) For each \( S_i \), it finds a vertex that has at least one neighbor outside \( S_i \). We call such vertices \textit{active} vertices. Whether or not a vertex is active can be determined with a constant number of cut queries by computing \( \left| E(v, S_i) \right| \) for \( v \in S_i \). We go over each \( S_i \) looking for an active vertex; once we find an active vertex in \( S_i \) we move on to \( S_{i+1} \). The vertices that are discovered to be inactive are ignored for all future rounds of the algorithm.

2) Next we randomly bipartition the set of connected components and use Recover-\( k \)-From-All with constant \( k \) to learn, for each active vertices on one side, a neighbor on the other side. As in the case of Borůvka’s algorithm, we then combine the components which are connected by edges we discovered and reduce the number of components by a constant fraction.

Note that, across all iterations of step 1, we make at most \( n \) many useless queries (i.e., queries where we find a vertex to be inactive). So we only need to account for the query complexity of step 2. Here we crucially use the fact that we can reduce the number of connected components by a constant factor to show that total number of cut queries required over all invocations of Recover-\( k \)-From-All is bounded by \( O(n) \).

The next task is to extend the spanning forest algorithm to also construct sparse edge connectivity certificates. For our application we will want to construct a sparse edge connectivity certificate of the contracted graph \( G' \), which is an integer weighted graph with \( q \in o(n) \) vertices. The most natural idea would be to extend our spanning forest algorithm to construct a spanning forest of such a graph while making only \( O(q) \) cut queries. We could then directly construct a sparse \( r \)-edge connectivity certificate with \( O(qr) \) cut queries by iteratively finding one spanning forest at a time. Unfortunately, we do not know how to find a spanning forest in a contracted graph more efficiently. The reason is that if the adjacency matrix of the graph has entries with magnitude \( M \) this introduces an extra \( \log(M + 1) \) factor into the separating matrix bounds, which we cannot afford.
Instead we revisit the spanning forest algorithm and further parallelize Borůvka’s algorithm by simultaneously building the different forests of the sparse edge connectivity certificate.\(^1\) We also crucially make use of the fact that \(G'\) is not an arbitrary integer weighted graph, but the contraction of a simple graph \(G\), and that for our application we can afford an extra additive \(O(n)\) term. The fact that \(G'\) is a contraction of a simple graph allows us to keep using the separating matrix machinery on Boolean matrices by working on appropriate submatrices of the adjacency matrix of \(G\), and the extra \(O(n)\) term is used to identify these submatrices.

Let \(F_1, \ldots, F_r\) be the spanning forests that we want to compute, recalling that \(F_1\) is spanning forest in the graph \(G \setminus (\bigcup_{j < i} F_i)\). Initially, each \(F_i\) is empty. As before, we use steps 1. and 2. to find edges to extend these spanning tree. However, the crucial difference is the following: We find these edges with respect to the connected components of the last tree \(F_r\), and we add each of these edges into the spanning forest \(F_i\) for the least value of \(i\) where it does not create a cycle. It is not hard to see that the set of connected components \(\{S^{(i)}_1, \ldots, S^{(i)}_{r+1}\}\) of \(F_{i+1}\) for different \(i \in [r]\) form a laminar family: \(\{S^{(i+1)}_1, \ldots, S^{(i+1)}_{r+1}\}\) is a refinement of \(\{S^{(i)}_1, \ldots, S^{(i)}_{r+1}\}\). However, we cannot expect that the number of connected components of \(F_r\) will decrease by a constant factor in each round as before. We can however show that it happens within \(O(r)\) rounds. This, together with a similar accounting of cut queries as before, leads to the following theorem.

**Theorem 2.5 (Informal version, see [24] for formal treatment):** Let \(G = (V, E)\) be an \(n\)-vertex simple graph, and let \(G' = (V', E')\) be a contraction of \(G\) with \(q\) superfices, for \(q\) sufficiently large. There is a zero-error randomized algorithm that makes \(O(n + rq\log(n)/\log(q))\) cut queries in expectation and outputs a sparse \(r\)-edge connectivity certificate for \(G'\).

**D. Open questions**

Our work raises some open questions that concern both upper bounds and lower bounds.

- **Lower bounds:** The tightness of a number of our algorithms hinges on a positive answer to the following questions.
  - **Communication complexity:** Can we show a lower bound of \(\Omega(n \log n)\) for the randomized two-party communication complexity of edge connectivity? The current best known lower bound in the randomized case is \(\Omega(n \log \log n)\) [22], while the deterministic communication complexity of this problem is known to be \(\Omega(n \log n)\) [14]. A positive answer to this question implies an \(\Omega(n)\) lower bound on the randomized cut query complexity of edge connectivity, showing that Theorem 1.1 is tight. On the flip side, a randomized algorithm for edge connectivity making \(o(n)\) cut queries would imply a negative answer to this question. It is reasonable to think that a lower bound of \(\Omega(n \log n)\) for the randomized two-party communication complexity should hold even for the simpler problem of deciding if graph is connected, and we conjecture this to be true. Proving this would resolve the randomized communication complexity of connectivity, which has remained open since the work of Babai, Frankl, and Simon [16].
  - **Minimum degree:** Does computing the minimum degree of a simple graph indeed require \(\Omega(\sqrt{n})\) quantum cut queries? As mentioned before, quantum minimum finding gives a simple \(O(\sqrt{n})\) upper bound. By a reduction from minimum degree to edge connectivity (Appendix of [24]), a positive answer would imply that our \(O(\sqrt{n})\) quantum cut query algorithm for edge connectivity is tight (up to polylogarithmic factors).
    - **Upper bounds:** Our algorithms could give rise to new algorithms and upper bounds in a number of ways.
      - **Weighted graphs:** Can we find a minimum cut in a weighted graph with \(O(n)\) cut queries? This would not violate any currently known lower bound, and it would improve on the \(O(n \log n)\) cut query algorithm from [18]. Similarly, can we find a minimum cut in a weighted graph with \(o(n)\) quantum cut queries or matrix-vector queries?
      - **(Approximate) edge connectivity with \(\text{polylog}(n)\) queries:** We mentioned that a key bottleneck for edge connectivity with quantum cut queries is computing the minimum degree, which might require \(\Omega(\sqrt{n})\) quantum cut queries. In contrast, we can approximate the minimum degree with \(\text{polylog}(n)\) quantum cut queries [32].

\(^1\)Nagamochi-Ibaraki [45] also construct all spanning forests of the sparse edge connectivity certificate in parallel. They iterate over each edge of the graph and place it in the first spanning forest in which it does not create a cycle. We cannot afford to iterate over all edges and instead modify Borůvka’s algorithm to build the spanning forests in parallel.

**REFERENCES**

[1] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, ser. Algorithms and Combinatorics. Springer, 1988, vol. 2.

[2] S. T. McCormick, “Submodular function minimization,” in *Discrete Optimization*, ser. Handbooks in Operations Research and Management Science. Elsevier, 2005, vol. 12, pp. 321–391. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927050705120076

[3] A. Schrijver, “A combinatorial algorithm minimizing submodular functions in strongly polynomial time,” *Journal of Combinatorial Theory, Series B*, vol. 80, no. 2, pp. 346–355, 2000. [Online]. Available: https://doi.org/10.1006/jctb.2000.1989
[42] C. Wulff-Nilsen, “Fully-dynamic minimum spanning forest with improved worst-case update time,” in Proceedings of the 49th Annual ACM-SIGACT Symposium on Theory of Computing (STOC ’17), 2017, pp. 1130–1143. [Online]. Available: http://doi.acm.org/10.1145/3055399.3055415

[43] T. Saranurak, “A simple deterministic algorithm for edge connectivity,” in Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA ’21), SIAM, 2021, pp. 80–85.

[44] P. Gawrychowski, S. Mozes, and O. Weimann, “Minimum cut in $O(m \log^2 n)$ time,” in Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP ’20), ser. LIPIcs, vol. 168. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 57:1–57:15.

[45] H. Nagamochi and T. Ibaraki, “A linear-time algorithm for finding a sparse $k$-connected spanning subgraph of a $k$-connected graph,” Algorithmica, vol. 7, no. 5&6, pp. 583–596, 1992. [Online]. Available: https://doi.org/10.1007/BF01758778

[46] V. Grebinski and G. Kucherov, “Reconstructing a Hamiltonian cycle by querying the graph: Application to DNA physical mapping,” Discrete Applied Mathematics, vol. 88, no. 1-3, pp. 147–165, 1998. [Online]. Available: https://doi.org/10.1016/S0166-218X(98)00070-5

[47] P. Beame, S. Har-Peled, S. N. Ramamouorthy, C. Rashtchian, and M. Sinha, “Edge estimation with independent set oracles,” ACM Transactions on Algorithms, vol. 16, no. 4, pp. 52:1–52:27, 2020.

[48] G. Ivanyos, H. Klauck, T. Lee, M. Santha, and R. de Wolf, “New bounds on the classical and quantum communication complexity of some graph properties,” in Proceedings of the 32nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS ’12), ser. LIPIcs, vol. 18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012, pp. 148–159.

[49] H. Jowhari, M. Saglam, and G. Tardos, “Tight bounds for $L_p$ samplers, finding duplicates in streams, and related problems,” in Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS ’11). ACM, 2011, pp. 49–58.