Quantum Speedup for Graph Sparsification, Cut Approximation and Laplacian Solving

Simon Apers[*] Ronald de Wolf[†]

November 19, 2019

Abstract

Graph sparsification underlies a large number of algorithms, ranging from approximation algorithms for cut problems to solvers for linear systems in the graph Laplacian. In its strongest form, "spectral sparsification" reduces the number of edges to near-linear in the number of nodes, while approximately preserving the cut and spectral structure of the graph. The breakthrough work by Benczúr and Karger (STOC'96) and Spielman and Teng (STOC'04) showed that sparsification can be done optimally in time near-linear in the number of edges of the original graph.

In this work we show that quantum algorithms allow to speed up spectral sparsification, and thereby many of the derived algorithms. Given adjacency-list access to a weighted graph with n nodes and m edges, our algorithm outputs an ϵ-spectral sparsifier in time $\widetilde{O}(\sqrt{mn}/\epsilon)$. We prove that this is tight up to polylog-factors. The algorithm builds on a string of existing results, most notably sparsification algorithms by Spielman and Srivastava (STOC'08) and Koutis and Xu (TOPC'16), a spanner construction by Thorup and Zwick (STOC'01), a single-source shortest-paths quantum algorithm by Dürr et al. (ICALP'04) and an efficient k-wise independent hash construction by Christiani, Pagh and Thorup (STOC'15). Combining our sparsification algorithm with existing classical algorithms yields the first quantum speedup, roughly from $\widetilde{O}(m)$ to $\widetilde{O}(\sqrt{mn})$, for approximating the max cut, min cut, min st-cut, sparsest cut and balanced separator of a graph. Combining our algorithm with a classical Laplacian solver, we demonstrate a similar speedup for Laplacian solving, for approximating effective resistances, cover times and eigenvalues of the Laplacian, and for spectral clustering.

[*] Inria, France and CWI, the Netherlands. Supported by the CWI-Inria International Lab. `simon.apers@inria.fr`

[†] QuSoft, CWI and University of Amsterdam, the Netherlands. Partially supported by the Dutch Research Council (NWO) through Gravitation-grant Quantum Software Consortium - 024.003.037, and through QuantERA project QuantAlgo 680-91-034. `rdewolf@cwi.nl`

1 Introduction and Summary

1.1 Graph Sparsification

The complexity of many graph problems naturally scales with the number of edges in the graph. Graph sparsification aims to reduce this number of edges, while preserving certain quantities of interest. When considering for instance the approximation of cut problems such as MIN CUT or SPARSEST CUT, the aim is to sparsify the graph while approximately preserving its cut values. This was first shown to be possible in the pioneering work of Karger [Kar94] and later Benczúr and Karger [BK96]. They introduced the concept of *cut sparsifiers*, which are reweighted subgraphs that ϵ-approximate all cuts in the graph. We can then solve cut problems in the hopefully sparser subgraph, yielding an approximate solution to the original problem. Quite surprisingly, they showed that for any undirected graph with n nodes and m edges, there always exists a cut sparsifier with as few as $\widetilde{O}(n/\epsilon^2)$ edges, and moreover this sparsifier can be constructed in time $\widetilde{O}(m)$. This result lies at the basis of $\widetilde{O}(m)$-time approximation algorithms for amongst others MIN CUT [Kar94], MIN *st*-CUT [She13, KLOS14, Pen16], SPARSEST CUT and BALANCED SEPARATOR [ARV09, She13]. We refer the interested reader to [PQ82, Shm97] for surveys on the many applications of cut approximation.

In their breakthrough work on Laplacian solvers, Spielman and Teng [ST11] strengthened the notion of cut sparsifiers to so-called *spectral sparsifiers*. Rather than preserving the cut structure, these reweighted subgraphs preserve the spectral structure or *quadratic form* of the Laplacian associated to the graph. More specifically, H is an ϵ-spectral sparsifier of G if

$$(1-\epsilon)L_G \preceq L_H \preceq (1+\epsilon)L_G,$$

with L_H and L_G the Laplacian matrices associated to H resp. G. Since the value of any cut can be expressed as a quadratic form in the Laplacian, any spectral sparsifier is necessarily a cut sparsifier. More importantly it implies that Laplacian systems, which are linear systems in the graph Laplacian, can be approximately solved using the Laplacian of the sparsified graph. Similar to the case for cut sparsifiers, Spielman and Teng showed the existence and $\widetilde{O}(m)$-time construction of ϵ-spectral sparsifiers with $\widetilde{O}(n/\epsilon^2)$ edges. This formed a critical cornerstone of their $\widetilde{O}(m)$-time solver for Laplacian systems, and the string of results and algorithms that followed it - commonly referred to as the "Laplacian paradigm" [Ten10]. Some examples among these are faster algorithms for learning [ZGL03, ZHS05], computer vision and image processing [KMT11], spectral clustering [Vis13, OSV12] and computing random walk properties [CKP$^+$16]. The sparsification results of Spielman and Teng were later refined most notably by Spielman and Srivastava [SS11] and Batson, Spielman and Srivastava [BSS12]. In [BSS12], the existence of spectral sparsifiers with only $O(n/\epsilon^2)$ edges was proved, which later inspired the resolution of the famous Kadison-Singer problem by Marcus, Spielman and Srivastava [MSS15].

1.2 Main Result and Applications

In this work we propose a quantum algorithm for spectral sparsification, leading to the theorem below. The algorithm builds on a range of quantum and classical results, the most important of which are classical sparsification algorithms by Spielman and Srivastava [SS11] and Koutis and Xu [KMP14], a spanner algorithm by Thorup and Zwick [TZ05], a quantum algorithm for single-source shortest-path trees by Dürr, Heiligman, Høyer and Mhalla [DHHM06] and an efficient k-independent hash function by Christiani, Pagh and Thorup [CPT15].

Theorem 1. *There exists a quantum algorithm that, given adjacency-list access to a weighted and undirected graph G, outputs with high probability the explicit description of an ϵ-spectral sparsifier of G with $\widetilde{O}(n/\epsilon^2)$ edges in time $\widetilde{O}(\sqrt{mn}/\epsilon)$.*

Note that sparsification is only useful when $n/\epsilon^2 \in O(m)$ and hence $\epsilon \geq \sqrt{n/m}$. This implies that $\widetilde{O}(\sqrt{mn}/\epsilon) \in \widetilde{O}(m)$, and hence our algorithm provides a quantum speedup over classical algorithms, whose $\widetilde{O}(m)$ runtime can be shown to be optimal. For dense graphs, having $m \in \Omega(n^2)$, this improves the time complexity from $\widetilde{O}(n^2)$ classically to $\widetilde{O}(n^{3/2})$. The algorithm outputs an explicit description of the sparsifier, and has a space requirement of $O(\log n)$ qubits and $\widetilde{O}(\sqrt{mn}/\epsilon)$ classical bits. We prove a matching lower bound, showing that the runtime of our algorithm is optimal up to polylog-factors. In fact, we show that even outputting a weaker *cut* sparsifier requires the same amount of queries.

Theorem 2. *Any quantum algorithm that, given adjacency-list access to a weighted and undirected graph G, explicitly constructs with high probability an ϵ-cut sparsifier of G has query complexity $\widetilde{\Omega}(\sqrt{mn}/\epsilon)$.*

Our algorithm provides a direct speedup for many of the aforementioned applications. In Table 1 we illustrate this speedup for a number of cut approximation problems. All bounds follow by combining our sparsification algorithm with the best classical algorithms, applied to the sparsifier. As far as we know, this is the first quantum speedup for these cut approximation problems.

	Classical	Quantum (this work)
.878-MAX CUT	$\widetilde{O}(m)$ [AK16]	$\widetilde{O}(\sqrt{mn})$
ϵ-MIN CUT	$\widetilde{O}(m)$ [Kar00]	$\widetilde{O}(\sqrt{mn}/\epsilon)$
ϵ-MIN st-CUT	$\widetilde{O}(m + n/\epsilon^5)$ [Pen16]	$\widetilde{O}(\sqrt{mn}/\epsilon + n/\epsilon^5)$
$O(\sqrt{\log n})$-SPARSEST CUT/BAL.SEP.	$\widetilde{O}(m + n^{1+\delta})$ [She09]	$\widetilde{O}(\sqrt{mn} + n^{1+\delta})$

Table 1: Classical and quantum time complexity of cut approximation problems. All quantum bounds follow from combining our quantum sparsification algorithm with the corresponding classical algorithm. Parameter δ is an arbitrarily small but positive constant.

We can also use a classical Laplacian solver on the sparsifier to find a speedup for Laplacian solving, i.e., solving the linear system $Lx = b$ where L is the Laplacian of the original graph.

Theorem 3 (Quantum Laplacian Solver)**.** *There exists a quantum algorithm that, given adjacency-list access to a weighted and undirected graph G, outputs with high probability an approximate solution $\tilde{x}$ to the linear system $Lx = b$ such that $\|\tilde{x} - x\|_L \leq \epsilon \|x\|_L$ in time $\widetilde{O}(\sqrt{mn}/\epsilon)$.*

Here $\|v\|_L$ denotes the L-induced norm $\|v\|_L = \sqrt{v^\dagger L v} = \|L^{1/2} v\|$, with $v^\dagger$ the complex transpose of vector v. This is the typical norm considered for Laplacian solving. This speeds up the dependency on m with respect to classical solvers [ST14], whose runtime is $\widetilde{O}(m \log(1/\epsilon))$. We also find quantum speedups for approximating effective resistances and random walk commute times, creating an approximate "resistance oracle" which allows to query for the effective resistance of *any* node pair in time $\widetilde{O}(1)$, approximating the random walk cover time, and approximating the bottom eigenvalues of the Laplacian. Finally we discuss how a spectral sparsifier allows to implement spectral k-means clustering more efficiently, so that our quantum sparsification algorithm also leads to a speedup for this task. We summarize our speedups in Table 2, and discuss prior work on quantum algorithms for some of these problems in Section 1.5.

	Classical	Quantum (this work)
ϵ-Laplacian Solving	$\widetilde{O}(m)$ [ST14]	$\widetilde{O}(\sqrt{mn}/\epsilon)$
ϵ-Effective Resistance (single)	$\widetilde{O}(m)$	$\widetilde{O}(\sqrt{mn}/\epsilon)$
ϵ-Effective Resistances (all)	$\widetilde{O}(m + n/\epsilon^4)$ [SS11]	$\widetilde{O}(\sqrt{mn}/\epsilon + n/\epsilon^4)$
$O(1)$-Cover Time	$\widetilde{O}(m)$ [DLP11]	$\widetilde{O}(\sqrt{mn})$
k bottom eigenvalues	$\widetilde{O}(m + kn/\epsilon^2)$	$\widetilde{O}(\sqrt{mn}/\epsilon + kn/\epsilon^2)$
Spectral (k-means) Clustering	$\widetilde{O}(m + n \operatorname{poly}(k))$	$\widetilde{O}(\sqrt{mn} + n \operatorname{poly}(k))$

Table 2: Classical and quantum time complexity of Laplacian solving and some of its applications. All classical bounds without reference follow from [ST14]. All quantum bounds follow from combining our quantum sparsification algorithm with the corresponding classical algorithm.

1.3 Quantum Algorithm

Our quantum sparsification algorithm starts from the iterative sparsification algorithm by Koutis and Xu [KX16]. Their algorithm provides a simple combinatorial counterpart to the usual, algebraic treatment of spectral sparsification. It crucially relies on the growth of so-called *spanners* of the graph, which are sparse subgraphs that approximately preserve all pairwise distances between nodes. After growing a small number of disjoint spanners in the graph, and keeping these edges, they downsample the remaining edge set by keeping every edge independently with some fixed constant probability, and discarding the rest. This results in a sparsifier with approximately half the number of edges of the original graph. Repeating this procedure a logarithmic number of times results in an ϵ-spectral sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges.

The gist of our quantum speedup comes from a faster quantum algorithm for constructing spanners. This algorithm follows essentially by pairing a classical spanner algorithm by Thorup and Zwick [TZ05] with the shortest-paths quantum algorithm by Dürr, Heiligman, Høyer and Mhalla [DHHM06]. More specifically we prove the theorem below, where we call a graph H a spanner of G if it is a subgraph with $O(n \log n)$ edges, and the distance between any pair of nodes in H is at most $\log n$ times their original distance in G. Our algorithm speeds up the classical $\widetilde{O}(m)$-time algorithm by Thorup and Zwick, whose runtime is optimal.

Theorem 4. *There exists a quantum algorithm that, given adjacency-list access to a weighted and undirected graph G, outputs with high probability a spanner of G in time $\widetilde{O}(\sqrt{mn})$.*

We can now try to plug this faster spanner construction in the Koutis-Xu sparsification algorithm. The problem, however, is that we cannot output the "intermediate" sparsifiers, since after a constant number of iterations these still have $\Omega(m)$ edges, whereas we aim for a runtime that scales with $\sqrt{mn}$. We overcome this issue using two observations, which will allow us to describe the intermediate graphs only implicitly.

First we show that *if* we were given query access to a uniformly random "advice string" of $\widetilde{O}(m)$ bits, then we could implicitly mark the discarded edges, and grow spanners in the remaining, unmarked graph without significantly affecting the runtime. Second, we get rid of this long advice string by using that any $(k/2)$-step quantum algorithm cannot distinguish a uniformly random string from a k-wise independent string, which only behaves uniformly random for subsets of at most k elements. This is a known result and can be proven for instance using the polynomial method [BBC$^+$01]. Hence it suffices that we have access to a k-wise independent random string, allowing us to use the rich literature on *k-independent hash functions* that aim to simulate access to such random strings. Specifically we require the recent result by Christiani, Pagh and Thorup [CPT15], which shows that in $\widetilde{O}(k)$ time we can construct a data structure that can simulate queries to a k-wise independent string, requiring only $\widetilde{O}(1)$ time per query. Prior to their work, all algorithms required preprocessing time $\widetilde{O}(k^{1+\delta})$, for $\delta > 0$. Using their construction we can efficiently simulate the random advice string, which leads to the following claim.

Claim 1. *Consider any quantum algorithm with runtime q that uses a uniformly random advice string. Then we can construct a quantum algorithm without advice string that has the same output distribution and has a runtime $\widetilde{O}(q)$.*

Combining these observations remedies the issue of having to store the intermediate graphs, and leads to a speedup of the Koutis-Xu algorithm runtime down to time $\widetilde{O}(\sqrt{mn}/\epsilon^2)$.

We then further improve the runtime down to $\widetilde{O}(\sqrt{mn}/\epsilon)$ by combining this quantum sparsification algorithm with the sparsification toolbox of Spielman and Srivastava [SS11]. In that work, they show that a graph can be sparsified very elegantly by sampling edges with weights roughly proportional to their effective resistances. Complementing this, they propose a near-linear time constructible "resistance oracle", which allows to query for effective resistances in logarithmic time. We use our quantum sparsification algorithm to construct an initial, rough sparsifier with a constant error, in time $\widetilde{O}(\sqrt{mn})$. We then construct an approximate resistance oracle for this sparsifier, which effectively yields an approximate resistance oracle for the original graph. Surprisingly, such rough approximation suffices for constructing an ϵ-spectral sparsifier using the Spielman-Srivastava sampling scheme. This finally allows us to sample the $\widetilde{O}(n/\epsilon^2)$ edges of the sparsifier in time $\widetilde{O}(\sqrt{mn}/\epsilon)$. This idea of using a "poor" spectral approximation to compute sampling probabilities to obtain a better spectral approximation is also used in [LMP13, CLM$^+$15].

1.4 Matching Lower Bound

We prove that the $\widetilde{O}(\sqrt{mn}/\epsilon)$-runtime of our quantum algorithm is optimal, up to polylog-factors, even when we wish to construct a weaker *cut* sparsifier. The intuition behind this is that an ϵ-cut sparsifier of a general graph must contain $\Omega(n/\epsilon^2)$ edges (and this is tight [BSS12]). If we can appropriately "hide" these edges among the m edges of a graph, then a quantum search algorithm requires $\Theta(\sqrt{mn/\epsilon^2}) = \Theta(\sqrt{mn}/\epsilon)$ queries to retrieve them.

Turning this intuition into a concrete lower bound, however, turns out to be rather complicated. We start with a random graph construction by Andoni, Chen, Krauthgamer, Qin, Woodruff and Zhang [ACK$^+$16]. This construction describes graphs on n nodes and $\widetilde{O}(n/\epsilon^2)$ edges, so that any ϵ-cut sparsifier must contain a constant fraction of the edges. As such, the constructed graphs are in fact already sparsifiers. We then carefully "hide" these sparsifiers in a larger, denser graph, in such a way that a sparsifier of this graph must retrieve all of the original, hidden sparsifiers. To prove a quantum lower bound for this search problem, we describe it as the composition of the problem of finding a constant fraction of the nonzero bits in a Boolean matrix with the OR-function. Finally we combine lower bounds for the individual problems using a composition theorem for adversary bounds, applicable to the composition of a relational problem with a function. This composition theorem was very recently proven by Belovs and Lee [BL19], prompted by our question to them.

1.5 Prior Work

We are not aware of any prior work on quantum speedups for graph sparsification. In a very different line of work though, spectral sparsification has been studied in a quantum context with the goal of sparsifying Hamiltonian matrices, which are used to describe many-body systems. Aharonov and Zhou [AZ19] asked whether the *interaction graph* of a many-body system can be sparsified while preserving its spectrum, showing that this is not possible in general. More recently, Herbert and Subramanian [HS19] considered the weaker notion of sparsifying the Hamiltonian matrix, and suggested that sparsification could indeed help in Hamiltonian simulation. They do not consider quantum algorithms for effectively constructing such a sparsifier.

Research on quantum algorithms for cut approximation is also limited. There is recent work by Hamoudi, Rebentrost, Rosmanis and Santha [HRRS19] on quantum approximate minimization of submodular functions, which can be used for cut approximation. However, their work was more recently superseded by better classical algorithms [ALS19]. Other recent work by Brandão, Kueng and Stilck França [BKSF19] used quantum SDP-solvers to approximate quadratic binary optimization problems, of which MAX CUT is the most notable instance. They do not succeed in finding a speedup for MAX CUT though, mainly because their algorithm does not benefit from the special structure of this instance.

Concerning our speedup for Laplacian solving, we mention a range of papers on quantum speedups for general linear system solving. Most famous is the work by Harrow, Hassidim and Lloyd [HHL09], which was later refined in work by Ambainis [Amb12] and Childs, Kothari and Somma [CKS17]. They describe a quantum algorithm for solving general linear systems $Ax = b$ in time $\widetilde{O}(d_M \kappa \log(1/\epsilon))$, with d_M the row sparsity and κ the condition number of A. These algorithms are particularly relevant for sparse and well-conditioned systems (in general, however, κ can be as large as $O(n^3 w_{\max}/w_{\min})$ for graph Laplacians [ST14, Lemma 6.1]). Crucially, they only output a quantum state that encodes the solution, rather than an explicit description as we do.

Prior to our work, quantum speedups have also been studied for the problems of estimating effective resistances and spectral gaps. Ito and Jeffery [IJ19] describe a quantum algorithm for estimating a single effective resistance $R_{s,t}$ in the adjacency matrix model. Their algorithm requires time $\widetilde{O}(n\sqrt{R_{s,t}}/\epsilon^{3/2})$, which is $\widetilde{O}(n^{3/2}/\epsilon^{3/2})$ in the worst case, and is restricted to graphs with unit edge weights. We improve[1] the worst-case complexity to $\widetilde{O}(\sqrt{mn}/\epsilon) \in \widetilde{O}(n^{3/2}/\epsilon)$, and our algorithm works for arbitrary edge weights. In addition, we effectively approximate *all* effective resistances simultaneously in the graph in that complexity. In different work by Wang [Wan17] and Chakraborty, Gilyén and Jeffery [CGJ19], quantum walks in the adjacency list model are used to estimate a single effective resistance. However, similar to quantum linear system solvers, these only find a quantum speedup for sparse graphs with a large spectral gap. In addition, their algorithms have a polynomial dependency on the maximum edge weight, whereas we only have a logarithmic dependence. A quantum walk algorithm for estimating the second bottom eigenvalue λ_2 of the Laplacian in the adjacency-matrix model was studied by Jarret, Jeffery, Kimmel and Piedrafita [JJKP18]. They give a multiplicative ϵ-approximation of λ_2 in time $\widetilde{O}(n/(\sqrt{\lambda_2}\epsilon))$, which is $\widetilde{O}(n^2/\epsilon)$ in the worst case. We improve the worst-case complexity to $\widetilde{O}(\sqrt{mn}/\epsilon) \in \widetilde{O}(n^{3/2}/\epsilon)$.

We also briefly mention some past work on quantum speedups for clustering. We could retrieve a single work by Daskin [Das17] describing a quantum algorithm for spectral clustering. In this work it is, however, mentioned explicitly that no direct speedup is found with respect to classical algorithms. Less directly related, there exists a number of papers [ABG07, LMR13, WKS15, KLLP18] on quantum speedups for k-means clustering and the construction of a neighborhood graph. These tasks are complementary to our work on finding a spectral embedding, given a similarity graph of the data. It does seem interesting to try and use these algorithms to further speed up our spectral

[1] We note that our algorithm works in the adjacency-list model. However, the adjacency-matrix model can be interpreted as an adjacency-list model with $m \in \Theta(n^2)$ edges. Hence, in the adjacency-matrix model our algorithm has complexity $\widetilde{O}(n^{3/2}/\epsilon)$, which still improves on [IJ19].

clustering algorithm.

Finally, we mention some classical work on sublinear algorithms for Laplacian solving and spectral sparsification. First, the work by Andoni, Krauthgamer and Pogrow [AKP19] describes a sublinear algorithm for Laplacian solving, with the aim of approximating a single coordinate of the output. Their algorithm is inspired by quantum algorithms for linear system solving, and similarly only finds a speedup for sparse and well-conditioned systems. The second work is by Lee [Lee13], who proposes a classical algorithm for spectral sparsification of unweighted graphs which is sublinear in m. He succeeds in bypassing the $\Omega(m)$ lower bound on classical sparsification by only achieving a weaker, additive error in the approximation. As such this work is incomparable to ours.

1.6 Open Questions

Our work raises a number of interesting questions and future directions, some of which we summarize below.

- The $\widetilde{O}(\sqrt{mn}/\epsilon)$ runtime of our Laplacian solver has an inversely linear dependence on the error parameter ϵ, whereas classical solvers have a logarithmic dependence on ϵ of their runtime $\widetilde{O}(m \log(1/\epsilon))$. We are convinced that this can also be achieved in the quantum case. This is in contrast to sparsification algorithms, both quantum and classical, whose polynomial dependence on ϵ is necessary.
- Work on Laplacian solvers has also led to progress on the long-standing question of computing maximum flows in graphs [CKM$^+$11, She13, KLOS14, Pen16], ultimately leading to classical algorithms with runtime $\widetilde{O}(m)$ that approximate maximum flows. Since the naive description of such a flow already requires time $\Omega(m)$, this seems optimal. We might, however, hope to find a quantum speedup for approximating certain quantities of the flow, or a compressed representation. A slower quantum algorithm for finding an *exact* maximum flow was already proposed by Ambainis and Špalek [AŠ06].
- At first sight, sparsifiers can only yield approximate solutions to cut problems. However, for the case of MIN CUT, Karger [Kar99, Kar00] has shown that in fact they can also be used to provide an *exact* solution in time $\widetilde{O}(m)$. We leave it as an open question whether our algorithm allows to speed up the exact MIN CUT problem.
- Spectral sparsification of graphs and Laplacians has been extended in different directions such as sparsification of hypergraphs [SY19, BST19], sparsification of sums of positive semi-definite matrices [SY19, SHS16], sparsification in a streaming setting [KL13, KLM$^+$17]. It is also closely related to concepts such as spectral sketching [ACK$^+$16] and linear data regression using leverage scores [DMM06]. It seems likely that we can also find quantum speedups for these related problems. Similarly we might hope to solve more "quantum" tasks, such as sparsifying density operators or POVMs.

2 Preliminaries

Throughout the paper we say that something holds "with high probability" if it holds with probability at least $1 - O(1/n)$.

2.1 Computational Model and Quantum Algorithms

As our computational model we assume a quantum-accessible classical control system that can run quantum subroutines on at most $O(\log n)$ qubits. We quantify the complexity of our algorithms by their runtime, which measures the total number of elementary operations or quantum gates that the algorithm requires.

We will only use qubits to implement Grover's quantum algorithm [Gro96] for searching sets of marked elements, which is summarized in the claim below.

Claim 2 (Repeated Grover Search)**.** *Let $f : [N] \to \{0,1\}$ be a function that marks a set of elements $S = \{i \in [N] \mid f(i) = 1\}$. Then there is a quantum algorithm that finds S with probability at least $2/3$ in $\widetilde{O}(\sqrt{N\,|S|})$ elementary operations and queries to f, and uses $O(\log N)$ qubits and $\widetilde{O}(|S|)$ classical bits.*

While we also use a quantum algorithms for finding shortest-path trees by Dürr, Heiligman, Høyer and Mhalla [DHHM06], the quantum routines in this algorithm can be reduced to Grover search.

2.2 Graphs, Queries and Spanners

We consider undirected, weighted graphs $G = (V, E, w)$ with $|V| = n$ nodes and $|E| = m$ edges, and edge weights $w : E \to \mathbb{R}_{\geq 0}$. We are given *adjacency-list access* to G, as is considered in e.g. [DHHM06, GR02]. This allows to query for the degree of a node, its k-th neighbor (according to some unknown but fixed ordering), or the weight of an edge. This model is more restrictive than both the "general graph model", which in addition allows for adjacency matrix queries [Gol10], and the "sparse-access model", in which the neighbors are ordered lexicographically, as is commonly assumed in quantum algorithms for linear system solving and Hamiltonian simulation [HHL09, CKS17].

We define the *distance* $\delta_G(u, v)$ between nodes u and v with respect to G as

$$\delta_G(u,v) = \min_{u-v \text{ path } P} \sum_{e \in P} \frac{1}{w_e}.$$

This definition is in accordance with the interpretation of G as an electrical network, in which an edge e corresponds to a link of conductance w_e (and hence resistance or "cost" $1/w_e$), as is common in the literature on spectral sparsification. A spanner of G is a sparse subgraph H that approximately preserves all pairwise distances. Specifically, we will call H a *t-spanner* of G if for any pair $u, v \in V$ it holds that

$$\delta_G(u,v) \leq \delta_H(u,v) \leq t\delta_G(u,v).$$

Note that the first inequality is trivially satisfied since H is a subgraph. It is well-known that every weighted graph has a $(2k-1)$-spanner with $O(n^{1+1/k})$ edges [ADD$^+$93]. Throughout the paper we will use the shorthand *spanner* to denote a t-spanner with $t = 2\log n$ and $\widetilde{O}(n)$ edges. An *r-packing of spanners* of G is an ordered set $H = (H_1, H_2, \ldots, H_r)$ of r edge-disjoint spanners such that H_j is a spanner for $G - \cup_{i<j} H_i$, which is the remaining graph after removal of the edges of all previous spanners.

The *Laplacian* L of a weighted graph G is given by $L = D - A$, with A the weighted adjacency matrix $(A_{ij}) = w_{ij}$ and D the diagonal weighted degree matrix $(D_{ii}) = \sum_j w_{ij}$. Alternatively, we can rewrite the Laplacian as

$$L = \sum_{e \in E} w_e \chi_e \chi_e^T,$$

where we let $\chi_e = \chi_u - \chi_v$ denote a vector associated to the edge $e = (u, v)$, with χ_u, χ_v indicator vectors of the nodes u, v (we fix an arbitrary orientation of the edges). If G is connected then L_G has a trivial kernel consisting only of the all-ones vector. Moreover, L_G is a real, symmetric, diagonally dominant matrix with nonnegative diagonal entries, and is hence positive semi-definite.

2.3 Spectral Sparsification using Spanner Packings

A *cut sparsifier* H of a graph G is a sparse, reweighted subgraph that preserves the value of all cuts. Specifically, H is called an ϵ-cut sparsifier if for any $S \subseteq V$ it holds that

$$(1-\epsilon)\mathrm{val}_G(S) \leq \mathrm{val}_H(S) \leq (1+\epsilon)\mathrm{val}_G(S), \tag{1}$$

where $\mathrm{val}_G(S) = \sum_{i \in S, j \notin S} w_{(i,j)}$ denotes the total weight of the edges leaving S.

A *spectral sparsifier* H of a graph G is a sparse, reweighted subgraph that preserves the quadratic form $x^T L_G x$ associated to the Laplacian L_G of G, for any vector $x \in \mathbb{C}^n$. Specifically, H is called an ϵ-spectral sparsifier if for any $x \in \mathbb{C}^n$ it holds that

$$(1-\epsilon)x^T L_G x \leq x^T L_H x \leq (1+\epsilon)x^T L_G x. \tag{2}$$

Alternatively, we can rewrite this as $(1-\epsilon)L_G \preceq L_H \preceq (1+\epsilon)L_G$, where $A \preceq B$ denotes that $A - B$ is positive semi-definite. This condition implies for instance that all eigenvalues of H ϵ-approximate the eigenvalues of G [BSST13], and all cuts in H ϵ-approximate those in G. To see the latter, consider a subset $S \subseteq V$ and let χ_S denote the indicator on S, then

$$\chi_S^T L_G \chi_S = \sum_{(u,v)=e \in E} w_e (\chi_S(u) - \chi_S(v))^2 = \mathrm{val}_G(S).$$

This shows that the cut value can be described by a quadratic form in the Laplacian, and hence (2) implies that $(1-\epsilon)\mathrm{val}_G(S) \leq \mathrm{val}_H(S) \leq (1+\epsilon)\mathrm{val}_G(S)$, for all $S \subseteq V$. Any ϵ-spectral sparsifier is therefore also an ϵ-cut sparsifier.

Spectral sparsifiers can be constructed by using spanners to identify the "important" edges in the graph. This was first noticed by Kapralov and Panigrahy [KP12], and further refined by Koutis and Xu [KX16]. We will build on the latter work, which describes a very elegant approach for constructing spectral sparsifiers from spanner packings. Their algorithm iteratively invokes the routine described below, which creates a spectral sparsifier with approximately half the number of edges of the original graph.

Algorithm 1 $H = \textbf{Half Sparsify}(G, \epsilon)$

```
construct an $O(\log^2(n)/\epsilon^2)$-packing of spanners of $G$
let $P$ be their union and set $H = P$
for each edge $e \notin P$ do
    with probability 1/4, add $e$ to $H$ with weight $4w_e$
return $H$
```

Theorem 5 ([KX16, Theorem 3.2])**.** *The graph $H = \textbf{Half Sparsify}(G, \epsilon)$ is, with probability at least $1 - 1/n^2$, an ϵ-spectral sparsifier of G with at most $m/2 + \widetilde{O}(n/\epsilon^2)$ edges.*

Iterating this routine $O(\log(m/n))$ times yields with high probability an ϵ-spectral sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges, which is optimal up to log-factors [ACK$^+$16]. Classically the complexity is dominated by the construction of $\widetilde{O}(1/\epsilon^2)$ spanners, each of which requires time $\widetilde{O}(m)$ [TZ05], giving a total time complexity $\widetilde{O}(m/\epsilon^2)$.

3 Quantum Sparsification Algorithm

In this section we describe our quantum algorithm for constructing spectral sparsifiers. The algorithm is based on the scheme by Koutis and Xu. We use as a black-box a quantum algorithm for constructing a spanner in time $\widetilde{O}(\sqrt{mn})$, whose description we postpone to Section 5.

As we already mentioned in the introduction, we cannot simply plug this quantum spanner algorithm in the Koutis and Xu algorithm. Indeed, after a single iteration of their algorithm this would require to output a graph with up to $m/2$ edges, which is much too costly since we aim at a runtime that scales as $\sqrt{mn}$. We resolve this issue in two stages. First, we assume that we have access to a random "advice string" of length $\widetilde{O}(m)$. We use this string to mark edges that have been discarded at some iteration by 0-bits, which we later use to implicitly set their weight equal to zero. By its construction, the spanner algorithm can then construct a spanner in the remaining graph. At the end we use Grover search to explicitly retrieve the remaining $\widetilde{O}(n/\epsilon^2)$ edges, whose union forms the spectral sparsifier. We then get rid of the random advice string. To this end we use efficient k-independent hash functions that allow to simulate queries to a k-wise independent random advice string. This suffices since by standard results a k-query quantum algorithm cannot distinguish a $2k$-wise independent advice strings from a uniformly random one.

3.1 Using Random Advice

We first assume access to a family of independent, random advice strings $r_i \in \{0,1\}^m$, with indices $i \in [\log(m/n)]$, such that all bits are independent and equal to 1 with probability 1/4. For different indices i, the strings r_i will function as consecutive "sieves" of the edge set.

Algorithm 2 describes the sparsification algorithm using such advice strings. A critical remark is that steps 4 and 5 of the algorithm are only performed implicitly, as mentioned before. Rather than keeping an explicit list of updated edge weights, we maintain an implicit "weight oracle". Only when an edge weight is queried, does this weight oracle calculate its weight by consulting the necessary advice strings. We show how to do this efficiently in the proof of Theorem 6.

Algorithm 2 $H = \textbf{Quantum Sparsify}(G, \epsilon)$

```
1: let $\{w'_e = w_e\}$ and $\ell = \lceil \log(m/n) \rceil$
2: for $i = 1, 2, \ldots, \ell$ do
3:     create an $O(\log^2(n)/\epsilon^2)$-packing of spanners of $G' = (V, E, w')$, let $P_i$ denote its union
4:     for each edge $e \notin P_i$ do                                          ▷ implicitly!
5:         if $r_i(e) = 1$ then set $w'_e = 4w'_e$ else set $w'_e = 0$
6: use repeated Grover search to find $H = \{e \in E \mid w'_e > 0\}$
```

Theorem 6. *Given access to independent, uniformly random advice strings $r_i \in \{0,1\}^m$, $i \in [\log(m/n)]$, algorithm* $\textbf{Quantum Sparsify}(G, \epsilon)$ *returns with probability $1 - O(\log(n)/n^2)$ an ϵ-spectral sparsifier of G with $\widetilde{O}(n/\epsilon^2)$ edges. There is a quantum algorithm that implements it in time $\widetilde{O}(\sqrt{mn}/\epsilon^2)$.*

Proof. Correctness easily follows from Theorem 5: in every iteration we "half-sparsify" the remaining graph (induced by all edges of weight $w_e > 0$). The probability that all $\log(m/n)$ iterations succeed is $1 - O(\log(n)/n^2)$. Below we discuss how steps 4 and 5 can be implemented efficiently, so that the runtime of the for-loop is dominated by the construction of $\widetilde{O}(1/\epsilon^2)$ spanners. By Theorem 13 this takes time $\widetilde{O}(\sqrt{mn}/\epsilon^2)$. By standard results [NC02], the repeated Grover search routine in the final step takes time $\widetilde{O}(\sqrt{mn}/\epsilon)$, which is the time needed to find n/ϵ^2 edges among m edges.

What remains to prove is that there exists an efficient oracle that keeps track of the weight updates in steps 4 and 5. Consider the i-th iteration. Given an edge e, let k denote the number of spanners before this iteration in which e occurs so far. If $k = 0$, return $w'_e = 4^i w_e$ if $(r_i\, r_{i-1} \dots r_1)(e) = 1$, and $w'_e = 0$ if $(r_i\, r_{i-1} \dots r_1)(e) = 0$. If $k > 0$, let $j < i$ denote the last spanner packing in which it occurs. Now return $w'_e = 4^{i-k} w_e$ if $(r_i\, r_{i-1} \dots r_{j+1})(e) = 1$, and $w'_e = 0$ otherwise. This takes $\widetilde{O}(1)$ searches through the set of spanners (which we may assume is sorted), and at most $O(i) \in \widetilde{O}(1)$ evaluations of the random oracle. □

The space complexity of the algorithm requires $O(\log n)$ qubits and $\widetilde{O}(n/\epsilon^2)$ classical bits. The number of qubits follows from the space complexity of the quantum spanner algorithm and the Grover search routine. The classical space complexity is dominated by the output size.

3.2 Using k-independent Hash Functions

In order to get rid of the random advice strings $\{r_i\}$, we build on the following fact, which is an easy consequence of the polynomial method [BBC$^+$01]. It seems that this was first used in the proof of [BFNR08, Theorem 19], and is stated explicitly in for instance [Zha15, Theorem 3.1].

Fact 1. *The output distribution of a quantum algorithm making q queries to a uniformly random advice string is identical to the same algorithm making q queries to a $2q$-wise independent advice string.*

As a consequence, we can replace the uniformly random advice strings of length m by a k-wise independent advice string with $k \in \widetilde{O}(\sqrt{mn}/\epsilon^2)$. Surely we also cannot explicitly construct a k-wise independent string of length $\widetilde{O}(m)$ in time $\widetilde{O}(\sqrt{mn}/\epsilon^2)$, but we can use hash functions to simulate queries to such a string. A family of hash functions $F = \{h : [u] \to [r]\}$ is called *k-independent* if, for any subset $S \subseteq [u]$ of size $|S| \leq k$ and a uniformly random function h in the family, the image of h on S behaves uniformly random in $[r]^{|S|}$. This implies that the image of a random member of F, which we will refer to as a *k-independent hash function*, describes a k-wise independent string over $[r]^u$. Elegant constructions of such functions have long been known, the most famous example being random degree-k polynomials, as proposed by Carter and Wegman [CW79]. Crucial to our cause, however, is that we can evaluate the hash function in $\widetilde{O}(1)$ time, potentially allowing $\widetilde{O}(k)$ preprocessing time. Fortunately, such a result was established very recently by Christiani, Pagh and Thorup [CPT15], who proved the theorem below. We note that this is a purely classical construction.

Theorem 7 ([CPT15])**.** *It is possible to construct in time $\widetilde{O}(k)$ a data structure of size $\widetilde{O}(k)$ that allows to simulate queries to a k-independent hash function in $\widetilde{O}(1)$ time per query.*

With $k = 2q$ and $[r] = \{0, 1\}$, we can combine this with Fact 1 to give the corollary below.

Corollary 1. *Consider any quantum algorithm with runtime q that makes queries to a uniformly random advice string. We can simulate this algorithm with a quantum algorithm with runtime $\widetilde{O}(q)$ without advice string, using $\widetilde{O}(q)$ additional classical bits.*

This shows that we can efficiently simulate the random advice string in Algorithm 2, leading to at most a polylogarithmic overhead in the runtime. The classical space complexity of the algorithm does increase from $\widetilde{O}(n/\epsilon^2)$ to $\widetilde{O}(\sqrt{mn}/\epsilon^2)$. The following theorem is immediate by combining Theorem 6 with Corollary 1.

Theorem 8. *There exists a quantum algorithm that, given adjacency-list access to a weighted and undirected graph G, constructs with high probability an ϵ-spectral sparsifier of G with $\widetilde{O}(n/\epsilon^2)$ edges in time $\widetilde{O}(\sqrt{mn}/\epsilon^2)$. The algorithm uses $O(\log n)$ qubits and $\widetilde{O}(\sqrt{mn}/\epsilon^2)$ classical bits.*

4 Refined Quantum Sparsification Algorithm

In the last section we proposed a quantum algorithm for constructing an ϵ-spectral sparsifier in time $\widetilde{O}(\sqrt{mn}/\epsilon^2)$. Here we show how to improve the runtime of this algorithm to $\widetilde{O}(\sqrt{mn}/\epsilon)$, which we will later show is optimal up to polylog-factors. The improvement essentially follows from combining our previous algorithm with the seminal results on spectral sparsification by Spielman and Srivastava [SS11]. In that work, they first showed that sampling edges with probabilities approximately proportional to their effective resistances results in a spectral sparsifier (the Koutis-Xu algorithm is derived from their result). Then they showed how Laplacian solvers could be used to efficiently estimate these effective resistances. We will use our quantum sparsification algorithm to first construct a "rough" ϵ-sparsifier, for some constant ϵ, which we only use to approximate the effective resistances in the original graph. Surprisingly such approximation suffices to implement the Spielman-Srivastava sampling scheme on the original graph. We then use a quantum sampling routine to efficiently implement this sampling scheme, finally leading to an ϵ-spectral sparsifier for arbitrary $\epsilon > 0$ in time $\widetilde{O}(\sqrt{mn}/\epsilon)$. This idea of using a "poor" spectral sparsifier for computing sampling probabilities to obtain a better spectral sparsifier is also present in for instance [LMP13, CLM$^+$15].

4.1 Spielman-Srivastava Toolbox and Quantum Sampling

Here we formally introduce the main tools that we use. These are an efficiently constructible "resistance oracle" and a sparsification algorithm based on this oracle from [SS11], and a quantum sampling routine for implementing this sparsification algorithm.

4.1.1 Approximate Resistance Oracle

The *effective resistance* in a graph G between a pair of nodes s and t is defined as the effective resistance between s and t after replacing every edge e by a resistor of value $1/w_e$. It can be expressed algebraically as $R_{s,t} = (\chi_s - \chi_t)^T L_G^+ (\chi_s - \chi_t)$, so that a Laplacian solver allows to efficiently compute $R_{s,t}$. Spielman and Srivastava proved that in some sense one can efficiently compute *all* effective resistances in roughly the same time. More specifically, they showed that it is possible to construct in near-linear time a data structure of size $\widetilde{O}(n/\epsilon^2)$ that allows to efficiently approximate $R_{s,t}$ for any s, t.

Theorem 9 ([SS11])**.** *Consider a weighted and undirected graph G. There is an $\widetilde{O}(m/\epsilon^2)$-time algorithm which computes a $(24\log(n)/\epsilon^2) \times n$ matrix Z such that with probability at least $1 - 1/n$, for every pair $s, t \in V$, it holds that*

$$(1-\epsilon)R_{s,t} \leq \|Z(\chi_s - \chi_t)\|^2 \leq (1+\epsilon)R_{s,t}.$$

Hence the matrix Z represents a data structure which allows to ϵ-approximate $R_{s,t}$ for any pair s, t by calculating the 2-norm distance between two columns, each of dimension $\widetilde{O}(1/\epsilon^2)$.

4.1.2 Spectral Sparsification with Edge Scores

In the same paper, Spielman and Srivastava proved the following theorem, which shows that a spectral sparsifier can be constructed by independently keeping edges with weights roughly proportional to their effective resistances.

Theorem 10 ([SS11]). *Let $2R_e \geq \widetilde{R}_e \geq R_e/2$ for each edge $e \in E$, and $p_e = \min(1, Cw_e\widetilde{R}_e \log(n)/\epsilon^2)$ for some universal constant C. Then keeping every edge e independently with probability p_e, and rescaling its weight with $1/p_e$, yields with probability at least $1 - 1/n$ an ϵ-spectral sparsifier of G with $O(n \log(n)/\epsilon^2)$ edges.*

Note that $\sum_e p_e \gg 1$ is the expected number of edges of the sparsifier. Since $\sum_e w_e R_e = n - 1$ [Bol13, Theorem 25], this yields the claimed number of edges.

We note that, in fact, Spielman and Srivastava describe a slightly different scheme. They propose to draw $\widetilde{O}(n/\epsilon^2)$ independent and identically distributed edge samples from the edge set, with probability proportional to their effective resistance. It is well-known that both schemes give the same performance bound - see e.g. [FHHP19, Remark 1].

4.1.3 Quantum Sampling

Assuming access to an approximate resistance oracle that gives approximations $\widetilde{R}_e$ to R_e, we wish to implement the Spielman-Srivastava sparsification scheme. While classically this requires time $\widetilde{O}(m + \sum_e p_e)$, we can use quantum algorithms to do so more efficiently.

Claim 3. *Assume we have query access to a list of probabilities $\{p_e\}_{e\in E}$. Then there is a quantum algorithm that samples a subset $S \subseteq E$, such that S contains every e independently with probability p_e, in expected time $\widetilde{O}(\sqrt{m(\sum_e p_e)})$.*

Proof. We can assume access to a random $\widetilde{O}(m)$-bit advice string r, since by Corollary 1 this implies that there also exists a quantum algorithm without random advice string. From this advice string we can derive a function $h_r : E \times [0, 1] \to \{0, 1\}$ such that for each e independently $h_r(e, p_e) = 1$ with probability p_e and $h_r(e, p_e) = 0$ otherwise. Combining this function with a query to the list of probabilities allows to implement an oracle $|e\rangle |0\rangle |0\rangle \mapsto |e\rangle |p_e\rangle |h_r(e, p_e)\rangle$. If T is the number of $e \in E$ for which $h_r(e, p_e) = 1$, then we can use repeated Grover search (Claim 2) to retrieve these in expected time $\widetilde{O}(\sqrt{mT})$. Since the expected value of T is $\sum_e p_e$, this proves the lemma. □

4.2 Refined Quantum Sparsification

Now we combine the Spielman-Srivastava toolbox, the quantum sampling routine and our quantum sparsification algorithm from the last section to improve the runtime of the latter from $\widetilde{O}(\sqrt{mn}/\epsilon^2)$ to $\widetilde{O}(\sqrt{mn}/\epsilon)$.

Algorithm 3 $H =$ **Quantum Sparsify**(G, ϵ)

```
use quantum sparsification (Theorem 8) to construct a (1/100)-spectral sparsifier $H$ of $G$
create a (1/100)-approximate resistance oracle of $H$ using Theorem 10, yielding estimates $\{\widetilde{R}_e\}$
use quantum sampling (Claim 3) to sample a subset of the edges, keeping every edge with
    probability $p_e = \min(1, Cw_e\widetilde{R}_e \log(n)/\epsilon^2)$
```

Theorem 11 (Quantum Spectral Sparsification). *Algorithm* **Quantum Sparsify**(G, ϵ) *returns with high probability an ϵ-spectral sparsifier H with $\widetilde{O}(n/\epsilon^2)$ edges, and has runtime $\widetilde{O}(\sqrt{mn}/\epsilon)$. The algorithm uses $O(\log n)$ qubits and $\widetilde{O}(\sqrt{mn}/\epsilon)$ classical bits.*

Proof. First we prove correctness. Since H is a spectral sparsifier of G, and effective resistances correspond to quadratic forms in the inverse of the Laplacian, we know that the effective resistances of H approximate those of G: $(1-1/100)R^G_{s,t} \leq R^H_{s,t} \leq (1+1/100)R^G_{s,t}$ for all $s, t \in V$. By Theorem 9 we know that the approximate resistance oracle yields estimates $\{\widetilde{R}^H_{s,t}\}$ such that $(1-1/100)R^H_{s,t} \leq \widetilde{R}^H_{s,t} \leq (1+1/100)R^H_{s,t}$. Combining these inequalities shows that

$$(1-1/100)^2 R^G_{s,t} \leq \widetilde{R}^H_{s,t} \leq (1+1/100)^2 R^G_{s,t}.$$

By Theorem 10, if we now keep every edge with probability $p_e = \min(1, C w_e \widetilde{R}^H_e \log(n)/\epsilon^2)$, then with probability $1-1/n$ we find an ϵ-spectral sparsifier with $O(n\log(n)/\epsilon^2)$ edges. Combining this success probability with those of the quantum sparsification algorithm and the construction of the resistance oracle, we find a total success probability of at least $(1-1/n)^3 = 1 - O(1/n)$.

The bound on the runtime follows from summing the $\widetilde{O}(\sqrt{mn})$ runtime of the quantum sparsification algorithm, the $\widetilde{O}(n)$ runtime for creating the resistance oracle of the sparsifier with $\widetilde{O}(n)$ edges, and the $\widetilde{O}(\sqrt{m(\sum_e p_e)})$ expected runtime of the quantum sampling routine. Since

$$\sum_e p_e \leq \frac{C\log(n)}{\epsilon^2} \sum_e w_e \widetilde{R}^H_e \leq \frac{(1+1/100)^2 C \log(n)}{\epsilon^2} \sum_e w_e R^G_e,$$

and $\sum_e w_e R^G_e = n-1$ [Bol13, Theorem 25], we have that $\sum_e p_e \in \widetilde{O}(n/\epsilon^2)$ and so the expected runtime of the sampling routine is $\widetilde{O}(\sqrt{mn}/\epsilon)$. Moreover, by the Chernoff bound the runtime of the latter routine will indeed be $\widetilde{O}(\sqrt{mn}/\epsilon)$ with probability at least $1-1/n$. Hence we can abort the algorithm whenever the runtime exceeds this bound, and the algorithm will still succeed with high probability, while the total runtime becomes $\widetilde{O}(\sqrt{mn}/\epsilon)$ in the worst case. □

5 Quantum Algorithm for Building Spanners

The Koutis-Xu sparsification algorithm identifies "important" edges by growing spanners inside the graph. In this section we propose a quantum algorithm for growing spanners, speeding up the best classical algorithms.

Recall from Section 2 that a t-spanner of a graph $G = (V, E, w)$ is a subgraph $H = (V, E_H \subseteq E, w)$ that preserves all pairwise distances between nodes up to a stretch factor t. For every pair $u, v \in V$, it should hold that

$$\delta_G(u,v) \leq \delta_H(u,v) \leq t\delta_G(u,v),$$

where we recall that $\delta_G(u,v) = \min_{u-v \text{ path } P} \sum_{e\in P} 1/w_e$. A spanner preserves the original weights on its edges. This is in contrast to spectral sparsifiers which are necessarily reweighted. A classic result by Althöfer et al. [ADD$^+$93] shows that, for any parameter $k > 0$, any n-node graph has a $(2k-1)$-spanner with $O(n^{1+1/k})$ edges. We refer the interested reader to the classic book by Peleg [Pel00] or the very recent survey by Ahmed et al. [ABS$^+$19].

There exists a range of classical algorithms for constructing spanners. We will make use of one by Thorup and Zwick [TZ05], which follows from their work on "approximate distance oracles". The main bottleneck of their algorithm is the growth of shortest-path trees in subgraphs. We speed up this bottleneck by using the quantum algorithm of Dürr, Heiligman, Høyer and Mhalla [DHHM06] for growing a shortest-path tree in time $\widetilde{O}(\sqrt{mn})$.

5.1 Thorup-Zwick Algorithm

The spanner algorithm from [TZ05] makes use of *shortest-path trees* (SPTs). A shortest-path tree $T(v)$ from a node v spanning a subset C is defined as a tree, rooted at v and spanning C, so that the distance in this tree from v to any node in C is the same as their distance in the original graph G.

The Thorup-Zwick algorithm, presented in Algorithm 4, randomly partitions the node set into k layers $\{A_i\}$, which are increasingly sparsified. The nodes in these layers function as "hubs" for the nearby nodes. Shortest-path trees are then grown that allow efficient routing along these hubs. The resulting spanner consists of the union of these shortest-path trees. In the algorithm below, we set $\delta(w, \emptyset) = \infty$ for any $w \in V$.

Algorithm 4 $H =$ **Spanner**(G, k)

```
let $A_0 = V$ and $A_k = \emptyset$
for $i = 1, 2, \ldots, k$ do
    if $i < k$, let $A_i$ contain each element of $A_{i-1}$, independently, with probability $n^{-1/k}$
    for $v \in A_{i-1} - A_i$ do
        grow shortest-path tree $T(v)$ from $v$ spanning $C(v) = \{w \in V \mid \delta(w, v) < \delta(w, A_i)\}$
        add $T(v)$ to $H$
```

Apart from the correctness of the algorithm, we will require some additional bounds on the size of the intermediate clusters $C(v)$. We extract the following theorem from the analysis by Thorup and Zwick.

Theorem 12 ([TZ05])**.**

- *The output graph H of* **Spanner**(G, k) *is a $(2k-1)$-spanner of G.*
- *The expected number of edges in H is $O(\mathbb{E}(\sum_v |C(v)|)) \in O(kn^{1+1/k})$.*
- *The expected number of edges with at least one node in the clusters is $\mathbb{E}(\sum_v |E(C(v))|) \in O(kmn^{1/k})$.*

Setting $k = 1/2 + \log n$, as we will do later on, this yields a $2\log n$-spanner with an expected number of edges $O(n \log n)$.

5.2 Quantum Spanner Algorithm

We can use a quantum algorithm from Dürr, Heiligman, Høyer and Mhalla [DHHM06] to speed up the construction of the shortest-path tree $T(v)$, spanning $C(v)$. We slightly generalize their algorithm to deal with "forbidden edges", which are encoded by associating a weight $w_e = 0$ to them (which corresponds to an infinite resistance or cost). Such edges will correspond to edges going outside of $C(v)$, as well as edges that have already been discarded by our sparsification algorithm.

In Appendix A we prove the following statement. We define the connected component of a node v_0 as the smallest subset $C_{v_0} \subseteq V$ such that $v_0 \in C_{v_0}$ and either $E(C_{v_0}, V \backslash C_{v_0}) = \emptyset$ or $\max\{w_e \mid e \in E(C_{v_0}, V \backslash C_{v_0})\} = 0$. This implies that there is no path of finite distance between v_0 and any node outside C_{v_0}.

Proposition 1. *Assume adjacency-list access to a weighted and undirected graph $G = (V, E, w)$. Let v_0 be a source node and C_{v_0} its connected component. Then there exists a quantum algorithm that outputs, with probability at least $1 - \delta$, a shortest-path tree from v_0 that spans C_{v_0}. It has a runtime $\widetilde{O}(\sqrt{|C_{v_0}||E(C_{v_0})|} \log(n/\delta))$ and requires $O(\log n)$ qubits and $\widetilde{O}(|C_{v_0}|)$ classical bits.*

From this we can speed up the spanner construction rather straightforwardly. To see this, note that the runtime of the Thorup-Zwick algorithm is dominated by the task of growing the shortest-path trees $T(v)$, spanning the local clusters $C(v)$, for all nodes $v \in V$. By setting $w_e = 0$ for any edge reaching out of $C(v)$, this task corresponds to a shortest-path tree on the connected component of v. If we use the above quantum algorithm to accelerate this, the total runtime becomes

$$\widetilde{O}\Big(\sum_v \sqrt{|C(v)||E(C(v)|}\Big) \in \widetilde{O}\Bigg(\sqrt{\sum_v |C(v)|}\sqrt{\sum_v |E(C(v)|}\Bigg),$$

where the containment follows from the Cauchy-Schwarz inequality. By Theorem 12 we know that $\mathbb{E}(\sum_v |C(v)|) \in O(kn^{1+1/k})$ and $\mathbb{E}(\sum_v |E(C(v))|) \in O(kmn^{1/k})$. By Markov's inequality this implies that with probability close to 1 the runtime is

$$\widetilde{O}\Big(\sqrt{kn^{1+1/k}}\sqrt{kmn^{1/k}}\Big) \in \widetilde{O}\big(kn^{1/k}\sqrt{mn}\big).$$

What remains to be shown is how we (implicitly) set $w_e = 0$ for all edges reaching out of $C(v)$. To that end we follow the idea of Thorup and Zwick of connecting a new source node s to every node in A_i, with edges of infinite weight, and construct an SPT from s to V. It is easy to see that this returns the shortest path from any node $w \notin A_i$ to A_i, allowing to calculate $\delta(w, A_i)$. Using the standard quantum SPT algorithm of [DHHM06] we can construct this SPT in time $\widetilde{O}(\sqrt{mn})$, and we do this whenever we construct a new A_i. Now assume that the quantum SPT algorithm at some point wishes to choose an edge (w, w'), with w part of the SPT constructed so far, and w' an adjacent node. Then by design this must be a cheapest border edge of the SPT constructed so far, and $\delta(v, w') = \delta(v, w) + \delta(w, w')$. Hence we know $\delta(v, w')$ and we can simply check whether $\delta(v, w') < \delta(w, A_i)$, setting the weight of the edge equal to zero if this is not the case. This proves the following theorem.

Theorem 13. *There exists a quantum algorithm that outputs in time $\widetilde{O}(kn^{1/k}\sqrt{mn})$ with high probability a $(2k-1)$-spanner of G of size $O(kn^{1+1/k})$. The algorithm uses $O(\log n)$ qubits and $\widetilde{O}(kn^{1+1/k})$ classical bits.*

Setting $k = \log n + 1/2$, we find an $\widetilde{O}(\sqrt{mn})$ quantum algorithm for constructing $2\log n$-spanners, as is required by our sparsification algorithm.

6 Matching Lower Bound: A Hidden Sparsifier

In this section we prove that the runtime of our quantum algorithm for spectral sparsification is optimal, up to polylog-factors. In fact, we show that even constructing a weaker *cut* sparsifier requires the same complexity. The following is a rephrasing of Theorem 2 from the introduction.

Theorem 14. *Fix n, m and $\epsilon \geq \sqrt{n/m}$. Consider the problem of outputting, with high probability, an explicit description of an ϵ-cut sparsifier of a weighted, undirected graph G with n nodes and m edges, given adjacency-list access to G. The quantum query complexity of this problem is $\widetilde{\Omega}(\sqrt{mn}/\epsilon)$.*

Note that sparsification is only meaningful under the constraint $\epsilon \geq \sqrt{n/m}$, since for $\epsilon \in O(\sqrt{n/m})$ the sparsifier would have at least as many edges as the original graph. We prove this lower bound by "hiding" a sparsifier in a larger graph, and then proving a quantum lower bound for finding the sparsifier. More specifically, we use a random graph construction by Andoni, Chen, Krauthgamer, Qin, Woodruff and Zhang [ACK$^+$16] such that any cut sparsifier must contain a constant fraction of the edges of the graph. We then hide a number of copies of this random graph by embedding it in a larger, denser graph. Finally we show that finding a constant fraction of the edges of the initial random graph requires $\widetilde{\Omega}(\sqrt{mn}/\epsilon)$ queries. To prove this lower bound, we combine a quantum lower bound for the OR-function with an information-theoretic lower bound for the problem of finding nonzero bits in a Boolean matrix. We can combine these separate lower bounds by using a composition theorem for adversary bounds, applicable to the composition of a relational problem with a function. Prompted by our question, such a composition theorem was very recently proven by Belovs and Lee [BL19].

6.1 Hiding a Sparsifier

We use a random graph construction of Andoni et al. [ACK$^+$16] for which a sparsifier must output a constant fraction of its edges. We then carefully hide a number of copies of this graph into a larger graph, which will later allow for the reduction of a query problem to the construction of a sparsifier.

6.1.1 An Unsparsifiable Graph

Andoni et al. [ACK$^+$16] construct a communication problem that is described by a family of random graphs on $2/\epsilon^2$ nodes with $1/(2\epsilon^4)$ edges. They show that for a constant fraction of the inputs ($> 3/5$), the communication problem requires to communicate $\Omega(1/\epsilon^4)$ bits. On the other hand, for at least a 2/3-fraction of the inputs, the communication problem can be solved by communicating an ϵ-cut sparsifier. This shows that for at least a $(3/5 - 1/3) > 1/4$-fraction of the inputs, the description of an ϵ-cut sparsifier requires $\Omega(1/\epsilon^4)$ bits. Using a slightly more refined argument, they even show that the number of edges of the sparsifier for these instances must be $\Omega(1/\epsilon^4)$.

Fix any $\epsilon > 0$. Let B_ϵ be any bipartite graph with $1/\epsilon^2$ nodes on each side, and every left node connected to a corresponding subset of half of the right nodes. From Andoni et al. [ACK$^+$16, Theorem 3.3] we can extract the following theorem. Indeed, if the claim would not hold, then we could violate their communication lower bound by communicating an ϵ-cut sparsifier.

Theorem 15 ([ACK$^+$16])**.** *For at least a $1/4$-fraction of all graphs B_ϵ, any ϵ-cut sparsifier must contain $\Omega(1/\epsilon^4)$ edges.*

It follows that at least a 1/4-fraction of all graphs B_ϵ cannot be significantly sparsified. Similarly to [ACK$^+$16], we will also consider larger families of disjoint copies of B_ϵ. We can easily prove the following corollary using the Chernoff bound.

Corollary 2. *Consider the disjoint union of ℓ distinct copies of B_ϵ. There exists a constant $\eta > 0$, independent of ℓ, such that for at least a $9/10$-fraction of all such graphs it holds that any ϵ-cut sparsifier must contain at least $\eta\ell/\epsilon^4$ edges.*

6.1.2 Embedding the Sparsifier

Fix n, $m \leq n^2/4$ and $\epsilon \geq \sqrt{n/m}$. Consider $\ell = \epsilon^2 n/2$ independent copies $B^{(k)}$ of B_ϵ, yielding a graph with n nodes. We wish to "hide" this graph in a larger, denser graph. To this end, we use an m-bit string x to (redundantly) describe the resulting graph, which we denote by $B(x)$. The description x consists of $\ell = \epsilon^2 n/2$ matrices $x^{(k)}$ of dimension $1/\epsilon^2 \times 1/\epsilon^2$,

$$x^{(k)} = \{x^{(k)}_{i,j} \mid i,j \in [1/\epsilon^2]\}, \quad k \in [\epsilon^2 n/2].$$

Every matrix $x^{(k)}$ is used to describe the bipartite adjacency matrix of the corresponding copy $B^{(k)}$. Rather than bits, however, the entries $x^{(k)}_{i,j}$ correspond to smaller strings of $N = 2\epsilon^2 m/n$ bits each, with at most one nonzero bit per string. We say that an input x is *valid* if it is of this form. The presence of an edge between the i-th left node and the j-th right node in $B^{(k)}$ is determined by the presence of a nonzero bit in the string $x^{(k)}_{i,j}$, i.e., by $\mathrm{OR}(x^{(k)}_{i,j})$. The bipartite adjacency list of the k-th copy $B^{(k)}$ is hence described as

$$\mathrm{adj}(B^{(k)}) = \mathrm{OR}(x^{(k)}) = \mathrm{OR}\left(\begin{bmatrix} 00010 & 00000 & 00000 & 10000 \\ 00100 & 00000 & 00100 & 00000 \\ 00000 & 01000 & 00001 & 00000 \\ 00000 & 00000 & 00010 & 01000 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

where we give concrete values to the bits in $x^{(k)}$ for illustration. In the next section we prove a lower bound on the identification of a constant fraction of the 1-bits in a valid input x. In this section we show how to embed the corresponding graph $B(x)$ in a larger, denser graph $G(x)$, so that an ϵ-cut sparsifier of $G(x)$ must identify a constant fraction of the edges of $B(x)$. This identifies a constant fraction of the 1-bits of the input x, so that the aforementioned lower bound effectively yields a lower bound for the construction of a sparsifier. We must take particular care in embedding $B(x)$ in $G(x)$ so as not to reveal additional information about x. E.g., we must prevent that the degrees of $G(x)$ reveal anything about the location of 1-bits in the input. We do this essentially by ensuring that a query to the adjacency list of $G(x)$ can be performed using a single query to x.

Initially, the oblivious (input-independent) mother graph $G = G_1 \cup G_2$ is the disjoint union of two bipartite graphs $G_1 = (L_1 \cup R_1, E_1)$ and $G_2 = (L_2 \cup R_2, E_2)$. The first graph G_1 is a multigraph consisting of $\ell = \epsilon^2 n/2$ disjoint copies $B^{(k)}$ of the *complete* bipartite graph on $1/\epsilon^2$ nodes, containing $N = 2\epsilon^2 m/n$ parallel copies of each edge (we will get rid of these multi-edges later). As such, G_1 has exactly n nodes and m (sometimes parallel) edges. We match every edge of G_1 to a unique input bit by matching the parallel copies of edge (i, j) in $B^{(k)}$ to the input bits in the string $x^{(k)}_{i,j}$. The second bipartite graph G_2 has at most n nodes and exactly m edges, but no multi-edges (we do not need to further specify this graph at this point). We formally match every single copy of an edge in G_1 to a unique edge in G_2, so that every input bit now corresponds to a unique edge in G_1 and a unique edge in G_2. While doing so, we ensure that all edges leaving a left or right node of G_1 are matched to edges in G_2 whose left (resp. right) ends are distinct. We will later clarify why this is important, and defer a proof that this is possible for some choice of G_2 to Appendix B. At this point, all edges in G have zero weight.

Next we take the input x into account, turning G into $G(x)$. To this end, we "flip" edge pairs conditioned on the input bit. Specifically, consider a bit $x(i)$ that corresponds to edges (l, r) in G_1 and (l', r') in G_2. If $x(i) = 1$, then we keep these edges as they are, except that we give (l, r) a unit weight. If $x(i) = 0$, then we "flip" the edges: we replace (l, r) and (l', r') by edges (l, l') and (r, r'). This is illustrated in Figure 1. We see now that if two outgoing edges from l were matched to edges with the same left end l' in L_2, then this could create the edge (l, l') twice. Similarly if two incoming edges from r were matched to the same right ends r', this could create the edge (r, r') twice. Our matching ensures that this can never happen.

Now consider a pair $l \in L_1$ and $r \in R_1$, corresponding to the edge (i, j) in $B^{(k)}$. Then $G(x)$ will contain a unique, unit-weight edge between l and r if and only if the string $x^{(k)}_{i,j}$ has a unique nonzero bit. As a consequence, $G(x)$ restricted to the node set $L_1 \cup R_1$ exactly describes $B(x)$. Moreover, we can perform a single query to the adjacency list of $G(x)$ using a single query to x, as we prove in the lemma below.

Lemma 1. *Fix n, $m \leq n^2/4$ and $\epsilon \geq \sqrt{n/m}$, and consider a valid input $x \in \{0,1\}^m$. Then $G(x)$ has at most $2n$ nodes and exactly $2m$ edges, and any query to its adjacency list can be simulated using a single query to x. $G(x)$ has $B(x)$ as a subgraph with unit edge weights, and all remaining edges have zero weight.*

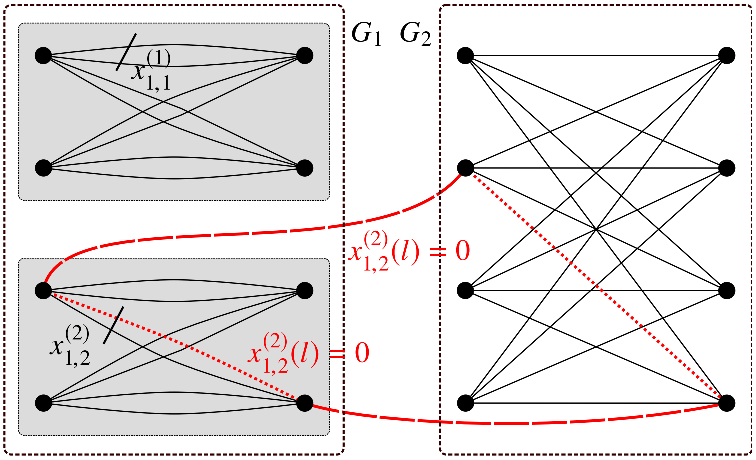

Figure 1: Matching input bits to edges in G_1 and G_2 for $n = 8$, $m = 16$ and $\epsilon = 1/\sqrt{2}$ (we set G_2 to be the complete bipartite graph for illustration only). The dotted red edges depict a pair of matched edges, which correspond to input bit $x^{(2)}_{1,2}(l)$. If $x^{(2)}_{1,2}(l) = 1$, these edges are kept in $G(x)$; if $x^{(2)}_{1,2}(l) = 0$, they are "flipped" with the dashed edges.

Proof. We only prove the claim about the query access, as all other claims follow easily from the construction. A degree query is trivially simulated, since $G(x)$ has the same degrees as G. To simulate a neighbor query, say that we wish to query the k-th entry of the adjacency list of node $l \in L_1$ in G_1, for some $k \in [2m/n]$. By construction, this corresponds to a known edge (l, r) in G_1, a corresponding edge (l', r') in G_2 and a unique input bit $x(i)$. We query the input bit $x(i)$. If it equals 1, then $G(x)$ has the unit-weight edge (l, r) and hence we return $r \in R_1$, and weight 1. If it equals 0, then we flipped the edges and $G(x)$ has the edge (l, l'), and so we return the node $l' \in L_1$ and weight 0. The same reasoning applies to queries from all other nodes of $G(x)$. □

We will also use the claim below, which follows easily from the fact that only the subgraph $B(x)$ of $G(x)$ has unit-weight edges, and all remaining edges have weight zero.

Claim 4. *Any ϵ-cut sparsifier of $G(x)$ contains an ϵ-cut sparsifier of $B(x)$.*

Proof. Let H be any ϵ-cut sparsifier of $G(x)$. Then we can simply remove all edges from H that are not in $B(x)$ (and necessarily have zero weight) to yield an ϵ-cut sparsifier for $B(x)$. □

Combining this claim with Corollary 2, we can deduce that for at least 9/10-th of all valid inputs x, any ϵ-cut sparsifier of $G(x)$ must identify a constant fraction (specifically, at least $\eta n/\epsilon^2$) of the edges of $B(x)$. Since the presence of an edge in $B(x)$, say (l_i, r_j) in copy $B^{(k)}$, reveals that string $x^{(k)}_{i,j}$ has a nonzero entry, we find the following corollary.

Corollary 3. *For at least a 9/10-fraction of all valid inputs x, it holds that any ϵ-cut sparsifier of $G(x)$ must identify a constant 2η-fraction of the nonzero strings.*

6.2 Finding the Sparsifier

Now we wish to prove a lower bound on identifying nonzero strings, thereby proving a lower bound on the complexity of sparsifying $G(x)$. To this end, we first formalize the new problem.

Definition 1. *The problem* $\mathrm{FindBits}_{r,c}$ *takes as input an* $r \times c$ *Boolean matrix, with each row containing exactly* $c/2$ *nonzero bits. A correct output consists of the indices of a* 2η*-fraction of the nonzero bits.*

We can prove an information-theoretic lower bound on the bounded-error quantum query complexity of this problem. This describes a lower bound on the number of queries required by a quantum algorithm that returns a correct output with probability at least 2/3.

Claim 5. *The bounded-error quantum query complexity of* $\mathrm{FindBits}_{r,c}$ *on any constant fraction of its valid inputs is* $\widetilde{\Omega}(rc)$*.*

Proof. This lower bound follows from combining an information-theoretic lower bound on how much information the algorithm extracts about its input, with Holevo's quantum information bound. More precisely, we examine the mutual information $I(A;B)$ between a uniformly random input matrix A, and an output B for $\mathrm{FindBits}_{r,c}$ that is correct with probability at least 2/3. We refer the reader to the textbook of Cover and Thomas [CT12] for precise definitions. By Holevo's theorem [Hol73], this mutual information lower bounds the quantum query complexity times $O(\log(rc))$.[2]

We lower bound $I(A;B) = H(A) - H(A|B)$ by lower bounding $H(A)$ and upper bounding $H(A|B)$. To lower bound $H(A)$, let S denote the set of all inputs, consisting of all $r \times c$ Boolean matrices with all row sums equal to $c/2$, and let $S' \subseteq S$ denote any constant fraction subset of S (say $|S'| = \beta|S|$ for some $\beta > 0$). Then A corresponds to a uniformly random element of S', and we have

$$H(A) = \log|S'| = \log(\beta|S|) = \log\left(\beta \prod_{j=1}^{r} \binom{c}{c/2}\right) = r\log\binom{c}{c/2} + \log\beta \geq r(c - O(\log c)).$$

To upper bound $H(A|B)$, we use that with probability at least 2/3 the output B is correct, revealing a 2η-fraction of the nonzero entries of A and hence significantly reducing the entropy of A. Let E denote an indicator bit which equals 1 if B is a correct output, and 0 otherwise. Then we can bound

$$\begin{aligned} H(A|B) &\leq H(A|B,E) + H(E) \\ &= \Pr(E=1)\, H(A|B,E=1) + \Pr(E=0)\, H(A|B,E=0) + H(E) \\ &\leq \frac{2}{3} H(A|B,E=1) + \frac{1}{3} rc + 1. \end{aligned}$$

We now want to upper bound $H(A|B, E=1)$. Fix integers $\{d(j)\}_{j\in[r]}$ such that $\sum_j d(j) = \eta rc$, and condition on the event $D_{\{d(j)\}}$ that B reveals $d(j)$ entries of the j-th row of A. Let $S_{\{d(j)\}} \subseteq S'$ denote the set of inputs which are compatible with $\{d(j)\}$. We can bound

$$\begin{aligned} H(A|B, E=1, D_{\{d(j)\}}) \leq \log|S_{\{d(j)\}}| &\leq \log\left(\prod_{j=1}^{r} \binom{c-d(j)}{c/2-d(j)}\right) \\ &= \sum_{j=1}^{r} \log\binom{c-d(j)}{c/2-d(j)} \leq \sum_{j=1}^{r} (c - d(j)) = r(1-\eta)c. \end{aligned}$$

[2] This is a fairly standard argument. We can consider a communication protocol where the T-query quantum algorithm implements each quantum query using one round of $O(\log(rc))$ qubits of communication to and from another party that holds A (by sending the query and answer registers back and forth), thus learning $I(A;B)$ bits of information about A while only communicating $O(T\log(rc))$ qubits. It now follows from [Hol73] that $T \in \Omega(I(A;B)/\log(rc))$.

This upper bound holds irrespective of the specific setting of $\{d(j)\}$. Hence, taking the expectation over all possible settings of $\{d(j)\}$, we obtain

$$H(A|B,E=1) = \mathbb{E}_{\{d(j)\}}\left[H(A|B,E=1,D_{\{d(j)\}})\right] \leq r(1-\eta)c.$$

Our lower bound on $H(A)$ and upper bound on $H(A|B)$ together imply

$$I(A;B) \geq \frac{2}{3}r(\eta c - O(\log c)) - 1 \in \Omega(rc). \qquad \square$$

We will also use the following lemma, which easily follows by using Grover's algorithm to check that indeed all output indices correspond to 1-bits in the $r \times c$ input matrix.

Lemma 2. *There exists a bounded-error quantum algorithm that, given quantum query access to an input x and an output y, verifies whether y is a correct output of* $\mathrm{FindBits}_{r,c}(x)$ *using $O(\sqrt{rc})$ quantum queries to x and y.*

Next we will compose $\mathrm{FindBits}_{r,c}$ with $r \times c$ copies of the OR_N-function on N bits each, so that every single input bit of $\mathrm{FindBits}_{r,c}$ is now described by the OR_N of N bits. We denote this composed problem as $\mathrm{FindBits}_{r,c} \circ \mathrm{OR}_N^{rc}$. To ensure that the input of the composed problem is a valid input for $\mathrm{FindBits}_{r,c}$, we must restrict it to $r \times c$ matrices of strings, each carrying N bits, so that exactly $c/2$ strings per row have one nonzero entry, and the remaining strings only have zeros.

The composed problem $\mathrm{FindBits}_{r,c} \circ \mathrm{OR}_N^{rc}$ corresponds to the composition of a relational problem (multiple outputs are correct for $\mathrm{FindBits}_{r,c}$) and a function. Bounds on the quantum query complexity of the composition of functions are well understood [HLS07]. When the outer problem is a relational problem, however, no such result seemed to be known. Prompted by our question, Belovs and Lee [BL19] very recently proved such a result, leading to the theorem below. We denote a relational problem by a set-valued function $f : S \subseteq \{0,1\}^M \to 2^T$, where S is a restricted set of inputs and T denotes its set of possible outputs. For $\mathrm{FindBits}_{r,c}$, each output in T corresponds to a set of ηrc indices of the $r \times c$ input matrix.

Theorem 16 ([BL19])**.** *Let $f : S \subseteq \{0,1\}^M \to 2^T$ be a relational problem whose bounded-error quantum query complexity is lower bounded by L. Assume that there is a bounded-error quantum algorithm that, given oracle access to $x \in S$ and some $a \in T$, verifies whether $a \in f(x)$ using $o(L)$ queries. Then the bounded-error quantum query complexity of the relational problem $f \circ \mathrm{OR}_N^M$, restricted to inputs x such that $\mathrm{OR}_N^M(x) \in S$, is lower bounded by $\Omega(L\sqrt{N})$.*

Now we let f denote the relational problem $\mathrm{FindBits}_{r,c}$, with S a $9/10$-fraction of its valid inputs. Then by Claim 5 its bounded-error quantum query complexity is $\widetilde{\Omega}(rc)$, and by Lemma 2 we can verify its output with bounded error using $O(\sqrt{rc})$ queries. Plugging this into the theorem above yields the following corollary.

Corollary 4. *Solving the problem* $\mathrm{FindBits}_{r,c} \circ \mathrm{OR}_N^{rc}$ *has bounded-error quantum query complexity $\widetilde{\Omega}(\sqrt{N}rc)$. This holds even when the inputs are restricted to a constant fraction of the valid inputs.*

Finally, using the graph embedding of the previous section, we show in the claim below how to solve this composed problem by constructing a cut sparsifier of an associated graph. Combining this with the lower bound from Corollary 4 then yields Theorem 14.

Claim 6. *Fix n, m, $\epsilon \geq \sqrt{n/m}$, and set $r = n/2$, $c = 1/\epsilon^2$ (number of potential neighbors) and $N = 2\epsilon^2 m/n$. For at least a $9/10$-fraction of all valid inputs x, we can reduce the problem* $(\mathrm{FindBits}_{r,c} \circ \mathrm{OR}_N^{rc})(x)$ *to finding an ϵ-cut sparsifier of $G(n,m,\epsilon,x)$. Specifically, any T-query quantum algorithm that sparsifies $G(n,m,\epsilon,x)$ with success probability at least $2/3$ can solve* $(\mathrm{FindBits}_{r,c} \circ \mathrm{OR}_N^{rc})(x)$ *with success probability at least $2/3$ on at least a $9/10$-fraction of its valid inputs using $O(T)$ queries.*

Proof. Consider a valid input $x \in \{0,1\}^m$ and the associated graph $G(x) = G(n, m, \epsilon, x)$ as defined in Section 6.1.2. For a 9/10-fraction of these inputs, we know by Corollary 3 that we can solve $(\mathrm{FindBits}_{r,c} \circ \mathrm{OR}_N^{rc})(x)$ by constructing an ϵ-cut sparsifier of $G(x)$. Moreover, by Lemma 1 we can query the adjacency list of $G(x)$ using a single query to x. Hence, if we output with probability at least 2/3 an ϵ-cut sparsifier of $G(x)$ using T queries to its adjacency list, then we also solve $(\mathrm{FindBits}_{r,c} \circ \mathrm{OR}_N^{rc})(x)$ with probability at least 2/3 using T queries to x. □

7 Applications

In this section we non-exhaustively list some applications of our quantum sparsification algorithm, proving speedups in cut approximation and Laplacian solving. Given the broad applicability of sparsification in classical algorithms, we expect that more applications will follow.

7.1 Cut Approximation

A range of cut approximation algorithms have a near-linear runtime in the number of edges in the graph. In the following we discuss a number of these, and show how our quantum algorithm for cut sparsification allows to speed them up.

7.1.1 MAX CUT

The MAX CUT problem for a weighted graph $G = (V, E, w)$ asks for a cut (S, S^c) that maximizes the total weight $\mathrm{val}_G(S)$ of the cut. Its decision version is one of the 21 problems famously shown to be NP-complete by Karp [Kar72]. The best current approximation factor of this problem is roughly .8785, as was famously proven by Goemans and Williamson [GW95] using an SDP relaxation. Khot et al. [KKMO07] showed that this approximation factor is optimal under the unique games conjecture.

In later work, Arora and Kale [AK16] showed how to solve the Goemans-Williamson SDP in time $\widetilde{O}(m)$, using amongst others the cut sparsification algorithm by Benczúr and Karger [BK96]. We find a quantum speedup by replacing the Benczúr-Karger algorithm by our quantum sparsification algorithm. Specifically we construct an ϵ-spectral sparsifier H, for some small but constant $\epsilon > 0$, in time $\widetilde{O}(\sqrt{mn}/\epsilon)$. Since all cuts are preserved up to a multiplicative error ϵ, we can apply the Arora-Kale algorithm on H to retrieve a MAX CUT approximation factor of at least $.8785(1-\epsilon)$. Choosing ϵ sufficiently small we find the following claim.

Claim 7. *There exists a quantum algorithm that outputs a .878-approximate max cut of a weighted graph in time $\widetilde{O}(\sqrt{mn})$.*

We note that for *unweighted* graphs, MAX CUT can be approximately solved classically in time $\widetilde{O}(n)$. If we wish to output the MAX CUT bipartition, then this is trivially optimal both for classical and quantum algorithms, and hence no quantum speedup is possible. The $\widetilde{O}(n)$ classical algorithm follows from the fact that for unweighted graphs a trivial sparsification procedure suffices to approximately preserve the MAX CUT value (pick $\widetilde{O}(n)$ edges uniformly at random - see for instance [Tre12, Section 2]). In the adjacency list model, this can be done classically in time $\widetilde{O}(n)$. For weighted graphs this approach no longer works, as in this case the edges have to be sampled with probability proportional to their edge weights. This cannot be done classically in time $o(m)$ since we could for instance hide a single heavy edge among m light edges. From all the cut problems we consider, this is the only one for which a classical sublinear algorithm with multiplicative error exists - albeit only for the unweighted case.

7.1.2 MIN CUT

Given a weighted graph $G = (V, E, w)$, the MIN CUT problem asks for a cut (S, S^c) with minimum weight $\mathrm{val}_G(S)$. Up to polylog-factors in the runtime, the current best algorithm is by Karger [Kar00]. It builds on cut sparsification to output an exact min cut of the graph with high probability in time $\widetilde{O}(m)$. Running this algorithm on our ϵ-cut sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges thus requires time $\widetilde{O}(n/\epsilon^2)$, and returns an ϵ-approximate min cut.

Claim 8. *There exists a quantum algorithm that outputs an ϵ-approximate min cut of a weighted graph in time $\widetilde{O}(\sqrt{mn}/\epsilon)$.*

We leave it as an open question whether this algorithm can be improved to also output an *exact* min cut, similar to the algorithm in [Kar00].

7.1.3 MIN *st*-CUT

Given a weighted graph $G = (V, E, w)$, the MIN *st*-CUT problem requires to output a cut $C = (S, S^c)$ with minimum value $\mathrm{val}_G(S)$, such that $s \in S$ and $t \notin S$. The current best algorithms for approximate MIN *st*-CUT build on the max-flow min-cut theorem, which states that the value of the min *st*-cut equals the value of a maximum *st*-flow. Combining classical cut sparsification with the recent $\widetilde{O}(m/\epsilon^3)$ solver for ϵ-approximate max flows by Peng [Pen16], this yields an $\widetilde{O}(m + n/\epsilon^5)$ time algorithm for MIN *st*-CUT. We obtain the following claim by running this algorithm on our cut sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges.

Claim 9. *There exists a quantum algorithm that outputs an ϵ-approximate minimum st-cut of a weighted graph in time $\widetilde{O}(\sqrt{mn}/\epsilon + n/\epsilon^5)$.*

7.1.4 SPARSEST CUT and BALANCED SEPARATOR

Given a weighted graph $G = (V, E, w)$, the SPARSEST CUT problem asks for a cut $C = (S, S^c)$ which minimizes the ratio $\mathrm{val}_G(S)/(|S|\,|S^c|)$. The BALANCED SEPARATOR problem asks in addition that the cut is "balanced", i.e., $\mu \leq |S|/|V| \leq 1/2$ for some constant $\mu > 0$. Exactly solving either of these problems is NP-hard, and optimally trying to approximate them has led to an interesting line of research [LR99, ARV09]. Currently the best polynomial-time classical algorithm for both problems achieves an $O(\sqrt{\log n})$-approximation factor in time $\widetilde{O}(m + n^{1+\delta})$, for an arbitrary positive constant δ. This is achieved by combining cut sparsification, work by Sherman [She09] which shows that an $O(\sqrt{\log n})$-approximation can be calculated by solving $\widetilde{O}(n^\delta)$ max flows, and the $\widetilde{O}(m/\epsilon^3)$ max flow algorithm by Peng [Pen16] for constant $\epsilon > 0$. Applying these algorithms to our ϵ-cut sparsifier for constant $\epsilon > 0$, we get the claim below.

Claim 10. *There exists a quantum algorithm that outputs an $O(\sqrt{\log n})$-approximate sparsest cut or balanced separator of a weighted graph in time $\widetilde{O}(\sqrt{mn} + n^{1+\delta})$, for an arbitrary constant $\delta > 0$.*

7.2 Quantum Laplacian Solver

The complexity of the best known linear system solver is $\widetilde{O}(n^\omega)$, with $\omega < 2.373$ the matrix multiplication coefficient. Building on a long line of work, Spielman and Teng [ST04] famously showed that the special case of Laplacian systems can be solved in time $\widetilde{O}(m) \in \widetilde{O}(n^2)$, with m the number of nonzero entries of the Laplacian. To this end they exploit the connection of Laplacians with graphs, allowing for a combinatorial interpretation and treatment of the linear system.

More specifically, a Laplacian solver aims to solve a linear system $L_G x = b$, where L_G is the Laplacian associated to some graph G. We denote the solution by $x = L_G^+ b$, where L_G^+ is the pseudoinverse of L_G (i.e., the inverse on its image). The original motivation for spectral sparsification was in fact to create better Laplacian solvers. Indeed, as follows from the lemma below, we can solve the Laplacian system in the sparsifier and retrieve an approximation of the original system. Here we use the A-induced norm $\|v\|_A = \sqrt{v^\dagger A v} = \|A^{1/2} v\|$ for a positive semi-definite matrix A, with $v^\dagger$ the Hermitian transpose of vector v.

Claim 11. *Consider a linear system $L_G x = b$, where L_G is the Laplacian of a weighted, undirected graph G. If H is an ϵ-spectral sparsifier of G, with Laplacian L_H, then solving $L_H x = b$ yields an approximate solution to the original system:*

$$\|L_H^+ b - x\|_{L_G} \leq 2\epsilon \|x\|_{L_G}.$$

Proof. Since H is an ϵ-spectral sparsifier of G, we have that $(1-\epsilon)L_G \preceq L_H \preceq (1+\epsilon)L_G$. This implies that the nonzero eigenvalues of $L_G L_H^+$ (and hence also of $L_G^{1/2} L_H^+ L_G^{1/2}$) lie between $1/(1+\epsilon)$ and $1/(1-\epsilon)$. With I the identity matrix restricted to the image of L_G and L_H, this implies that $\|L_G^{1/2} L_H^+ L_G^{1/2} - I\| \leq \epsilon/(1-\epsilon) \leq 2\epsilon$ for $\epsilon \leq 1/2$. From this we can bound

$$\begin{aligned} \|L_H^+ b - x\|_{L_G} &= \|L_G^{1/2} L_H^+ b - L_G^{1/2} x\| \\ &= \|L_G^{1/2} L_H^+ L_G x - L_G^{1/2} x\| \\ &= \|(L_G^{1/2} L_H^+ L_G^{1/2} - I) L_G^{1/2} x\| \leq \|L_G^{1/2} L_H^+ L_G^{1/2} - I\| \, \|L_G^{1/2} x\|. \end{aligned}$$

Since $\|L_G^{1/2} L_H^+ L_G^{1/2} - I\| \leq 2\epsilon$ and $\|L_G^{1/2} x\| = \|x\|_{L_G}$, this proves the lemma. □

This observation allowed Spielman and Teng to reduce the task of solving a potentially dense Laplacian system to solving a very sparse one in the Laplacian of the sparsifier, having $\widetilde{O}(n/\epsilon^2)$ edges. They then invoke other methods to efficiently solve the sparse Laplacian system in additional time $\widetilde{O}(n/\epsilon^2)$.

An immediate consequence is that we can use our quantum sparsification algorithm to speed up Laplacian solving. We can create a sparsifier H with $\widetilde{O}(n/\epsilon^2)$ edges in time $\widetilde{O}(\sqrt{mn}/\epsilon)$, and then use a classical Laplacian solver to solve the system $L_H x = b$ in an additional time $\widetilde{O}(n/\epsilon^2)$. This proves the proposition below.

Proposition 2 (Quantum Laplacian Solver)**.** *There exists a quantum algorithm that, given adjacency-list access to a weighted and undirected graph G, outputs with high probability an approximate solution $\tilde{x}$ to the linear system $L_G x = b$ such that $\|\tilde{x} - x\|_{L_G} \leq \epsilon \|x\|_{L_G}$ in time $\widetilde{O}(\sqrt{mn}/\epsilon)$.*

The use of the L_G-induced norm $\|\cdot\|_{L_G}$ is common in the study of Laplacian solvers. If, however, an ϵ-approximate solution in the regular 2-norm is desired, then one can use that $\|\tilde{x}-x\|_{L_G} \leq \epsilon\|x\|_{L_G}$ implies that $\|\tilde{x}-x\| \leq \epsilon\sqrt{\kappa}\|x\|$, with $\kappa = \lambda_{\max}/\lambda_{\min}$ the condition number of L_G. Setting $\epsilon = \delta/\sqrt{\kappa}$ hence yields a δ-approximation in the 2-norm, and the runtime of quantum Laplacian solver becomes $\widetilde{O}(\sqrt{mn\kappa}/\delta)$. This factor-$\sqrt{\kappa}$ overhead would typically be too large, apart from the case where the Laplacian is well-conditioned. A resolution to this issue follows from improving the error-dependence of our quantum Laplacian solver down to polylogarithmic, which is one of the open questions that we mentioned in the introduction.

7.2.1 Effective Resistances and Commute Times

Electrical networks, consisting of nodes $\{v \in V\}$ and resistors $\{r_e \,|\, e \in E\}$, are conveniently described by a weighted graph $G = (V, E, \{w_e = 1/r_e\})$ [Bol13]. Certain quantities of the electrical network can then be expressed using the Laplacian of G. One example is the effective resistance $R_{s,t}$ between a pair of nodes s and t, which we already encountered in Section 4. This can be expressed as a quadratic form in the inverse of the Laplacian:

$$R_{s,t} = (\chi_s - \chi_t)^T L^+ (\chi_s - \chi_t).$$

This effectively measures the *dissipated power* $\mathcal{E}(j_{s,t})$ of the electric flow that results from injecting a unit current in s, and extracting it from t, as is described by the demand vector $j_{s,t} = \chi_s - \chi_t$. For a more general demand vector $j \in \mathbb{R}^n$, with $1^T j = \sum_{v \in V} j(v) = 0$, the dissipated power is defined analogously:

$$\mathcal{E}(j) = j^T L^+ j.$$

Closely related to the effective resistance is the *commute time* of a random walk. The random walk commute time $C_{s,t}$ between nodes s and t is defined as the expected number of steps the walk must take from s to reach t, and then return to s. By a result of Chandra, Raghavan, Ruzzo, Smolensky and Tiwari [CRR$^+$96] we know that $C_{s,t} = 2WR_{s,t}$, with $W = \sum_e w_e$ the total edge weight in the graph.

Since the effective resistance and the dissipated power correspond to quadratic forms in the inverse of the Laplacian, they can be ϵ-approximated by calculating the corresponding quantity in an ϵ-spectral sparsifier. In addition, the total edge weight W_H of an ϵ-cut sparsifier ϵ-approximates the original edge weight W, so that $(1-\epsilon)^2 C^G_{s,t} \leq C^H_{s,t} \leq (1+\epsilon)^2 C^G_{s,t}$ and hence also the commute times are approximated. Using our quantum sparsification algorithm, together with a classical Laplacian solver, we can approximate any of these quantities in time $\widetilde{O}(\sqrt{mn}/\epsilon)$.

Claim 12. *Let $j \in \mathbb{R}^n$ be a current demand vector, with $1^T j = 0$. Then there exists a quantum algorithm that outputs an ϵ-approximation to the dissipated power $\mathcal{E}(j)$ in time $\widetilde{O}(\sqrt{mn}/\epsilon)$. In particular, if $j = \chi_s - \chi_t$ then this yields an ϵ-approximation of the effective resistance $R_{s,t}$ and the commute time $C_{s,t}$.*

By a similar argument, we can create an ϵ-spectral sparsifier, and construct the approximate resistance oracle of Spielman and Srivastava (see Section 4.1.1) on this sparsifier. By Theorem 9 we can construct this oracle in time $\widetilde{O}(n/\epsilon^4)$ (which corresponds to $\widetilde{O}(m/\epsilon^2)$ when we input a sparsifier with $m \in \widetilde{O}(n/\epsilon^2)$ edges). This proves the following claim.

Claim 13. *There exists an $\widetilde{O}(\sqrt{mn}/\epsilon + n/\epsilon^4)$-time quantum algorithm that outputs a $(24\log(n)/\epsilon^2) \times n$ matrix Z such that with high probability, for any $s, t \in V$ it holds that*

$$(1-\epsilon) R_{s,t} \leq \|Z(\chi_s - \chi_t)\|^2 \leq (1+\epsilon) R_{s,t}.$$

After creating the matrix Z, we can hence ϵ-approximate *any* effective resistance $R_{s,t}$ in time $\widetilde{O}(1/\epsilon^2)$, simply by calculating the norm of the difference between two $\widetilde{O}(1/\epsilon^2)$-length columns of Z. Apart from its use for spectral sparsification, as we demonstrated in Section 4, such an oracle can also be used to obtain geometric embeddings of the graph, see for instance [vL07].

7.2.2 Cover Time

The cover time $\tau_{\text{cov}}(G)$ of a weighted graph G denotes the expected number of steps before a random walk has visited all nodes, starting from the worst initial node. A classic bound on the cover time called *Matthew's bound* [Mat88] states that

$$\max_{s,t} H_{s,t} \leq \tau_{\text{cov}}(G) \leq (1 + \log(n)) \max_{s,t} H_{s,t},$$

with $H_{s,t}$ the hitting time from node s to node t. Tighter characterizations were later proven by Kahn, Kim, Lovász and Vu [KKLV00] and Ding, Lee and Peres [DLP11]. We extract the following claims from the latter.

Theorem 17. *[DLP11, Theorems 1.6 and 4.14]*

- *If H is an $O(1)$-spectral sparsifier of G, then $\tau_{\text{cov}}(H) \in \Theta(\tau_{\text{cov}}(G))$.*
- *There is an $\widetilde{O}(m)$ algorithm that outputs with high probability an $O(1)$-approximation of $\tau_{\text{cov}}(G)$.*

We can easily derive the following claim by using our quantum sparsification algorithm to construct an $O(1)$-spectral sparsifier, and then approximating the cover time on this sparsifier using the algorithm from [DLP11].

Claim 14. *Let G be an unweighted, undirected graph. There exists a quantum algorithm that outputs a constant-factor approximation to the cover time $\tau_{\text{cov}}(G)$ in time $\widetilde{O}(\sqrt{mn})$.*

7.2.3 Eigenvalues and Spectral Clustering

The bottom eigenvalues and eigenvectors of a graph Laplacian provide useful information about the graph, as is witnessed by e.g. Cheeger's inequality and spectral clustering [NJW02], the PageRank algorithm [BP98], and in fact the entire field of spectral graph theory [CDS80, Chu97]. Laplacian solvers allow to efficiently approximate these eigenvalues and eigenvectors. Spielman and Teng [ST14] (with a later refinement by Koutis, Levin and Peng [KLP16]) showed that in time $\widetilde{O}(m + kn/\epsilon^2)$ it is possible to compute (i) an ϵ-approximation to the k smallest eigenvalues $\lambda_1, \dots, \lambda_k$ of a Laplacian, and (ii) a set of k orthogonal unit vectors $v_1, \dots, v_k$ such that

$$v_\ell^T L v_\ell \leq (1 + \epsilon)\lambda_\ell, \quad 1 \leq \ell \leq k. \tag{3}$$

This set of vectors approximates the subspace spanned by the k bottom eigenvectors of the Laplacians. Already for constant $\epsilon > 0$, such a set can be used for spectral clustering. For the case of two clusters, this is explicitly discussed in [ST07] and [ST14, Section 7] for RatioCut. For k clusters, the most common approach is spectral k-means clustering [NJW02, vL07]. Using the same analysis as in [PSZ15], one can show that a set obeying (3) can be used to obtain the same performance [Zan19].

Using our quantum sparsification algorithm, we find a direct speedup for this task. To that end, note that it suffices to calculate the k smallest eigenvalues and approximate eigenvectors of an ϵ'-spectral sparsifier, for say $\epsilon' = \epsilon/10$. This will yield ϵ-approximate eigenvalues and eigenvectors of the original graph - see e.g. [ST14, Proposition 7.3]. We can hence construct an $(\epsilon/10)$-spectral sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges in time $\widetilde{O}(\sqrt{mn}/\epsilon)$, and then use a classical algorithm to solve the problem in the sparsifier, taking additional time $\widetilde{O}(kn/\epsilon^2)$. This proves the following claim.

Claim 15. *There exists an $\widetilde{O}(\sqrt{mn}/\epsilon + kn/\epsilon^2)$-time quantum algorithm that outputs with high probability an ϵ-approximation of each of the k smallest eigenvalues and a set of orthogonal unit vectors $v_1, \dots, v_k$ such that $v_\ell^T L v_\ell \leq (1 + \epsilon)\lambda_\ell$ for all $1 \leq \ell \leq k$.*

This directly yields a speedup for the aforementioned spectral clustering algorithms, provided that we are given adjacency-list access to some similarity graph of the data [vL07]. Consider for instance the spectral k-means clustering algorithm in [NJW02]. Given the similarity graph, its classical time complexity is dominated by (i) the time to construct a set of k vectors obeying (3) for constant $\epsilon > 0$, which is $\widetilde{O}(m + nk)$, and (ii) the time to perform k-means clustering on these vectors, which is $\widetilde{O}(n \operatorname{poly}(k))$. This yields a total classical complexity $\widetilde{O}(m + n \operatorname{poly}(k))$. From the above claim, we immediately find the following corollary.

Corollary 5. *There exists a quantum algorithm that, given adjacency-list access to the similarity graph of a data set, performs spectral k-means clustering on this data set in time $\widetilde{O}(\sqrt{mn} + n \operatorname{poly}(k))$.*

Acknowledgements

We are very grateful to Aleksandrs Belovs and Troy Lee for proving a composition property of the adversary method for relational problems, as we required for the proof of our lower bound. This work benefited from discussions with Joran van Apeldoorn, André Chailloux, Shantanav Chakraborty, András Gilyén, Anthony Leverrier, Christian Majenz, Christian Schaffner, Luca Trevisan and Luca Zanetti. During the completion of this work Simon Apers was supported by the CWI-Inria International Lab, and Ronald de Wolf was partially supported by the Dutch Research Council (NWO) through Gravitation-grant Quantum Software Consortium - 024.003.037 and through QuantERA project QuantAlgo 680-91-034.

References

[ABG07] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Quantum clustering algorithms. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 1–8. ACM, 2007. doi: 10.1145/1273496.1273497

[ABS$^+$19] Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. Graph spanners: A tutorial review. arXiv: 1909.03152, 2019.

[ACK$^+$16] Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P Woodruff, and Qin Zhang. On sketching quadratic forms. In *Proceedings of the 2016 Innovations in Theoretical Computer Science Conference (ITCS)*, pages 311–319. ACM, 2016. arXiv: 1511.06099

[ADD$^+$93] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993. doi: 10.1007/BF02189308

[AK16] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. *Journal of the ACM*, 63(2), 2016. doi: 10.1145/2837020

[AKP19] Alexandr Andoni, Robert Krauthgamer, and Yosef Pogrow. On solving linear systems in sublinear time. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 3:1–3:19. ACM, 2019. arXiv: 1809.02995

[ALS19] Brian Axelrod, Yang P Liu, and Aaron Sidford. Near-optimal approximate discrete and continuous submodular function minimization. arXiv: 1909.00171, 2019.

[Amb12] Andris Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *Proceedings of the 29th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 14, pages 636–647. LIPIcs, 2012. arXiv: `1010.4458`

[ARV09] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM*, 56(2):5, 2009. doi: `10.1145/1502793.1502794`

[AŠ06] Andris Ambainis and Robert Špalek. Quantum algorithms for matching and network flows. In *Proceedings of the 23rd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 172–183. Springer, 2006. arXiv: `quant-ph/0508205`

[AZ19] Dorit Aharonov and Leo Zhou. Hamiltonian sparsification and gap-simulation. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 2:1–2:21, 2019. arXiv: `1804.11084`

[BBC$^+$01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. arXiv: `quant-ph/9802049`

[BFNR08] Harry Buhrman, Lance Fortnow, Ilan Newman, and Hein Röhrig. Quantum property testing. *SIAM Journal on Computing*, 37(5):1387–1400, 2008.

[BK96] András A Benczúr and David R Karger. Approximating st minimum cuts in $\widetilde{O}(n^2)$ time. In *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC)*, volume 96, pages 47–55. ACM, 1996. doi: `10.1145/237814.237827`

[BKSF19] Fernando GSL Brandão, Richard Kueng, and Daniel Stilck França. Faster quantum and classical SDP approximations for quadratic binary optimization. arXiv: `1909.04613`, 2019.

[BL19] Aleksandrs Belovs and Troy Lee. More general adversary composition theorems. Forthcoming manuscript, 2019.

[Bol13] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 2013. doi: `10.1007/978-1-4612-0619-4`

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998. doi: `10.1016/S0169-7552(98)00110-X`

[BSS12] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012. arXiv: `0808.0163`

[BSST13] Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: theory and algorithms. *Communications of the ACM*, 56(8):87–94, 2013. doi: `10.1145/2492007.2492029`

[BST19] Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. arXiv: `1905.01495`, 2019.

[CDS80] Dragoš M Cvetkovic, Michael Doob, and Horst Sachs. *Spectra of graphs*, volume 10. Academic Press, New York, 1980. doi: `10.1002/zamm.19960760305`

[CGJ19] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 33:1–33:14, 2019. arXiv: `1804.01973`

[Chu97] Fan RK Chung. *Spectral graph theory.* Number 92. American Mathematical Society, 1997. doi: 10.1090/cbms/092

[CKM$^+$11] Paul Christiano, Jonathan A Kelner, Aleksander Madry, Daniel A Spielman, and Shang-Hua Teng. Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 273–282. ACM, 2011. arXiv: 1010.2921

[CKP$^+$16] Michael B Cohen, Jonathan Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. Faster algorithms for computing the stationary distribution, simulating random walks, and more. In *Proceedings of the 57th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 583–592. IEEE, 2016. arXiv: 1608.03270

[CKS17] Andrew M Childs, Robin Kothari, and Rolando D Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

[CLM$^+$15] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 6th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 181–190. ACM, 2015. arXiv: 1408.5099

[CPT15] Tobias Christiani, Rasmus Pagh, and Mikkel Thorup. From independence to expansion and back again. In *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC)*, pages 813–820. ACM, 2015. arXiv: 1506.03676

[CRR$^+$96] Ashok K Chandra, Prabhakar Raghavan, Walter L Ruzzo, Roman Smolensky, and Prasoon Tiwari. The electrical resistance of a graph captures its commute and cover times. *Computational Complexity*, 6(4):312–340, 1996. doi: 10.1007/BF01270385

[CT12] Thomas M Cover and Joy A Thomas. *Elements of information theory.* John Wiley & Sons, 2012.

[CW79] J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979. doi: 10.1016/0022-0000(79)90044-8

[Das17] Ammar Daskin. Quantum spectral clustering through a biased phase estimation algorithm. arXiv: 1703.05568, 2017.

[DHHM06] Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006. arXiv: quant-ph/0401091

[DLP11] Jian Ding, James R Lee, and Yuval Peres. Cover times, blanket times, and majorizing measures. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 61–70. ACM, 2011. doi: 10.4007/annals.2012.175.3.8

[DMM06] Petros Drineas, Michael W Mahoney, and Shan Muthukrishnan. Sampling algorithms for l_2 regression and applications. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1127–1136. Society for Industrial and Applied Mathematics, 2006.

[FHHP19] Wai-Shing Fung, Ramesh Hariharan, Nicholas JA Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. *SIAM Journal on Computing*, 48(4):1196–1223, 2019. doi: 10.1137/16M1091666

[Gol10] Oded Goldreich. Introduction to testing graph properties. In *Property testing*, pages 105–141. Springer, 2010.

[GR02] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi: `10.1007/s00453-001-0078-7`

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC)*, pages 212–219. ACM, 1996. arXiv: `quant-ph/9605043`

[GW95] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995. doi: `10.1145/227683.227684`

[HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009. arXiv: `0811.3171`

[HLS07] Peter Høyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC)*, pages 526–535. ACM, 2007. arXiv: `quant-ph/0611054`

[Hol73] Alexander S Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problemy Peredachi Informatsii*, 9(3):3–11, 1973. English translation in *Problems of Information Transmission*, 9:177–183, 1973.

[HRRS19] Yassine Hamoudi, Patrick Rebentrost, Ansis Rosmanis, and Miklos Santha. Quantum and classical algorithms for approximate submodular function minimization. arXiv: `1907.05378`, 2019.

[HS19] Steven Herbert and Sathyawageeswar Subramanian. Spectral sparsification of matrix inputs as a preprocessing step for quantum algorithms. arXiv: `1910.02861`, 2019.

[IJ19] Tsuyoshi Ito and Stacey Jeffery. Approximate span programs. *Algorithmica*, 81(6):2158–2195, 2019. arXiv: `1507.00432`

[JJKP18] Michael Jarret, Stacey Jeffery, Shelby Kimmel, and Alvaro Piedrafita. Quantum algorithms for connectivity and related problems. In *Proceedings of the 26th European Symposium on Algorithms (ESA)*, pages 49:1–49:13. Springer, 2018. arXiv: `1804.10591`

[Kar72] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972. doi: `10.1007/978-1-4684-2001-2_9`

[Kar94] David R Karger. Using randomized sparsification to approximate minimum cuts. In *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, volume 94, pages 424–432, 1994.

[Kar99] David R Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24(2):383–413, 1999. doi: `10.1287/moor.24.2.383`

[Kar00] David R Karger. Minimum cuts in near-linear time. *Journal of the ACM*, 47(1):46–76, 2000. arXiv: `cs/9812007`

[KKLV00] Jeff Kahn, Jeong Han Kim, Laszlo Lovasz, and Van H Vu. The cover time, the blanket time, and the Matthews bound. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 467–475. IEEE, 2000. arXiv: `math/0005121`

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007. doi: `10.1137/S0097539705447372`

[KL13] Jonathan A Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *Theory of Computing Systems*, 53(2):243–262, 2013. doi: `10.1007/s00224-012-9396-1`

[KLLP18] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. q-means: A quantum algorithm for unsupervised machine learning. arXiv: 1812.03584, 2018.

[KLM$^+$17] Michael Kapralov, Yin Tat Lee, CN Musco, Christopher Paul Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. *SIAM Journal on Computing*, 46(1):456–477, 2017. arXiv: 1407.1289

[KLOS14] Jonathan A Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 217–226. SIAM, 2014. arXiv: 1304.2338

[KLP16] Ioannis Koutis, Alex Levin, and Richard Peng. Faster spectral sparsification and numerical algorithms for SDD matrices. *ACM Transactions on Algorithms (TALG)*, 12(2):17, 2016. arXiv: 1209.5821

[KMP14] Ioannis Koutis, Gary L Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. *SIAM Journal on Computing*, 43(1):337–354, 2014. arXiv: 1003.2958

[KMT11] Ioannis Koutis, Gary L Miller, and David Tolliver. Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. *Computer Vision and Image Understanding*, 115(12):1638–1646, 2011. doi: 10.1007/978-3-642-10331-5_99

[KP12] Michael Kapralov and Rina Panigrahy. Spectral sparsification via random spanners. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*, pages 393–398. ACM, 2012. doi: 10.1145/2090236.2090267

[KX16] Ioannis Koutis and Shen Chen Xu. Simple parallel and distributed algorithms for spectral graph sparsification. *ACM Transactions on Parallel Computing (TOPC)*, 3(2):1–14, 2016. arXiv: 1402.3851

[Lee13] Yin Tat Lee. Probabilistic spectral sparsification in sublinear time. arXiv: 1401.0085, 2013.

[LMP13] Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. In *Proceedings of the 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 127–136. IEEE, 2013. arXiv: 1211.2713

[LMR13] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. arXiv: 1307.0411, 2013.

[LR99] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, 1999. doi: 10.1145/331524.331526

[Mat88] Peter Matthews. Covering problems for Markov chains. *The Annals of Probability*, 16(3):1215–1228, 1988. doi: 10.1214/aop/1176991686

[MSS15] Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison-Singer problem. *Annals of Mathematics*, pages 327–350, 2015. arXiv: 1306.3969

[NC02] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information.* Cambridge University Press, 2002.

[NJW02] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[OSV12] Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K Vishnoi. Approximating the exponential, the Lanczos method and an $\widetilde{O}(m)$-time spectral algorithm for balanced separator. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*, pages 1141–1160. ACM, 2012. arXiv: `1111.1491`

[Pel00] David Peleg. Distributed computing. *SIAM Monographs on discrete mathematics and applications*, 5, 2000. doi: `10.1137/1.9780898719772`

[Pen16] Richard Peng. Approximate undirected maximum flows in $O(m\,\mathrm{polylog}(n))$ time. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1862–1867. SIAM, 2016. arXiv: `1411.7631`

[PQ82] Jean-Claude Picard and Maurice Queyranne. Selected applications of minimum cuts in networks. *INFOR: Information Systems and Operational Research*, 20(4):394–422, 1982. doi: `10.1080/03155986.1982.11731876`

[PSZ15] Richard Peng, He Sun, and Luca Zanetti. Partitioning well-clustered graphs: Spectral clustering works! In *Conference on Learning Theory*, pages 1423–1455, 2015. arXiv: `1411.2021`

[She09] Jonah Sherman. Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cut. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 363–372. IEEE, 2009. arXiv: `0908.1379`

[She13] Jonah Sherman. Nearly maximum flows in nearly linear time. In *Proceedings of the 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 263–269. IEEE, 2013. arXiv: `1304.2077`

[Shm97] David B Shmoys. Approximation algorithms for np-hard problems. chapter Cut Problems and Their Application to Divide-and-conquer, pages 192–235. PWS Publishing Co., 1997.

[SHS16] Marcel K Silva, Nicholas JA Harvey, and Cristiane M Sato. Sparse sums of positive semidefinite matrices. *ACM Transactions on Algorithms (TALG)*, 12(1):9, 2016. arXiv: `1107.0088`

[SS11] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. arXiv: `0803.0929`

[ST04] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 81–90. ACM, 2004. arXiv: `cs/0310051`

[ST07] Daniel A Spielman and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. *Linear Algebra and its Applications*, 421(2-3):284–305, 2007. doi: `10.1016/j.laa.2006.07.020`

[ST11] Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011. arXiv: `0808.4134`

[ST14] Daniel A Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014. arXiv: `cs/0607105`

[SY19] Tasuku Soma and Yuichi Yoshida. Spectral sparsification of hypergraphs. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2570–2581. SIAM, 2019. arXiv: 1807.04974

[Ten10] Shang-Hua Teng. The Laplacian paradigm: Emerging algorithms for massive graphs. In *International Conference on Theory and Applications of Models of Computation*, pages 2–14. Springer, 2010. doi: 10.1007/978-3-642-13562-0_2

[Tre12] Luca Trevisan. Max cut and the smallest eigenvalue. *SIAM Journal on Computing*, 41(6):1769–1786, 2012. arXiv: 0806.1978

[TZ05] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, 2005. doi: 10.1145/1044731.1044732

[Vis13] Nisheeth K Vishnoi. Lx= b. *Foundations and Trends® in Theoretical Computer Science*, 8(1–2):1–141, 2013. doi: 10.1561/0400000054

[vL07] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. arXiv: 0711.0189

[Wan17] Guoming Wang. Efficient quantum algorithms for analyzing large sparse electrical networks. *Quantum Information and Computation*, 17(11-12):987–1026, 2017. arXiv: 1311.1851

[WKS15] Nathan Wiebe, Ashish Kapoor, and Krysta Svore. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Information and Computation*, 15(3&4):316–356, 2015. arXiv: 1401.2142

[Zan19] Luca Zanetti. Private communication, 2019.

[ZGL03] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 912–919, 2003.

[Zha15] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. *International Journal of Quantum Information*, 13(04):1550014, 2015. doi: 10.1007/978-3-642-32009-5_44

[ZHS05] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 1036–1043. ACM, 2005. doi: 10.1145/1102351.1102482

A Quantum Algorithm for Shortest-Path Trees

In the following we summarize the quantum algorithm for constructing a single-source shortest-path tree (SPT) by Dürr, Heiligman, Høyer and Mhalla [DHHM06]. We straightforwardly generalize it to the case where there is a set of "forbidden edges", which we encode by associating a weight $w_e = 0$ to these edges. The algorithm should then return an SPT over the connected component of the source. We recall that the resistance or "cost" of an edge is described by its resistance $1/w_e$, and the distance $\delta_G(u, v)$ between nodes u and v is

$$\delta_G(u,v) = \min_{u-v \text{ path } P} \sum_{e \in P} \frac{1}{w_e}.$$

The key routine that is used is a quantum algorithm for minimum-finding. It assumes oracle access to a "value function" $f : [N] \to \mathbb{N} \cup \{\infty\}$ and a "type function" $g : [N] \to \mathbb{N}$, with a total number of types

$$M = |\mathrm{Im}(g)| = |\{g(j) \,:\, j \in [N]\}|.$$

Given an integer $d \in \mathbb{N}$ with $d \leq N/2$, the problem $\textsc{minfind}(d, f, g)$ requires to output a subset $I \subseteq [N]$ of size $|I| = \min\{d, M\}$, such that

- $g(i) \neq g(j)$ for all distinct $i, j \in I$, and
- for all $j \in [N] \backslash I$ and $i \in I$, if $f(j) < f(i)$, then $f(i') \leq f(j)$ for some $i' \in I$ with $g(i') = g(j)$.

Proposition 3 ([DHHM06, Theorem 3.4])**.** *There is a quantum algorithm that solves* $\textsc{minfind}(d, f, g)$ *with success probability at least* $1 - \delta$ *in time* $\widetilde{O}(\sqrt{Nd}\log(1/\delta))$.

Algorithm 5 $T = \mathbf{SPT}(G, v_0)$

```
let $T = \{v_0\}$, $L = 1$ and $P_1 = \{v_0\}$
while $|T| < n$ do
    use MINFIND on $E(P_L)$ to construct a set $B_L$ of at most $|P_L|$ maximum-weight border edges
  from $P_L$ to distinct nodes
    let $(u, v)$ be a maximum-weight edge of $B_1 \cup \cdots \cup B_L$ with $v \notin P_1 \cup \cdots \cup P_L$.
    if $w(u, v) = 0$ then
        abort and output $T$
    else
        set $T = T \cup \{(u, v)\}$, $P_{L+1} = \{v\}$ and $L = L + 1$
    as long as $L \geq 2$ and $|P_L| = |P_{L-1}|$, merge $P_L$ into $P_{L-1}$ (removing $P_L$) and set $L = L - 1$
```

We define the *component* of a node v_0 as the smallest subset $C_{v_0} \subseteq V$ such that $v_0 \in C_{v_0}$ and either $|E(C_{v_0}, V \backslash C_{v_0})| = 0$ or $\max\{w_e \mid e \in E(C_{v_0}, V \backslash C_{v_0})\} = 0$.

Proposition 4. *Let* $G = (V, E, w)$ *be an undirected graph with weights* $w : E \to \mathbb{N}^*$*, and* C_{v_0} *the connected component of* $v_0 \in V$*. Then Algorithm* $\mathbf{SPT}(G, v_0)$ *outputs a shortest-path tree from* v_0 *that spans* C_{v_0}*. There is a quantum algorithm that implements* $\mathbf{SPT}(G, v_0)$ *with success probability* $1 - \delta$ *in time* $\widetilde{O}(\sqrt{|C_{v_0}||E(C_{v_0})|}\log(n/\delta))$.

Proof. We choose the success probability of each call to $\textsc{minfind}$ to be $1 - \delta/n$, so that the total success probability is at least $(1 - \delta/n)^n \geq 1 - \delta$. It was shown in [DHHM06] that in this case each iteration returns a maximum-weight border edge of T. Correctness of the algorithm hence follows from correctness of Dijkstra's algorithm.

The bound on the runtime follows from bounding the runtime of the calls to $\textsc{minfind}$, which are clearly dominant. At any iteration, we invoke $\textsc{minfind}(|P_L|, f, g)$ on $E(P_L)$, with

- type function g returning the end node $g((u, v)) = v$ of an edge (u, v), and
- value function f returning the inverse weight $1/w(u, v)$ of an edge (u, v) if $v \notin T$, and ∞ otherwise.

By Proposition 3 a single such call requires time $O(\sqrt{|E(P_L)||P_L|}\log(n/\delta))$. The total runtime is hence given by

$$O\bigg(\sum_{P_L} \sqrt{|E(P_L)||P_L|}\log(n/\delta)\bigg),$$

where the sum runs over the sets P_L in step 3 of each iteration. We can bound this by noting that the merging procedure in step 9 ensures that in every iteration $|P_L| = 2^{r_L}$ for some integer $r_L \leq \log n$, and any two P_L of the same size are necessarily disjoint. Since necessarily $P_L \subseteq C_{v_0}$, there are at most $|C_{v_0}|/2^r$ (disjoint) sets of size 2^r, and we can bound

$$\begin{aligned}\sum_{P_L : |P_L| = 2^r} \sqrt{|E(P_L)||P_L|} &= 2^{r/2} \sum_{P_L : |P_L| = 2^r} \sqrt{|E(P_L)|} \\ &\leq 2^{r/2} \sqrt{2^{-r}|C_{v_0}| \sum_{P_L : |P_L| = 2^r} |E(P_L)|} \leq \sqrt{2|C_{v_0}||E(C_{v_0})|},\end{aligned}$$

where the second inequality follows from Cauchy-Schwarz and the third inequality from the fact that $\sum_{P_L : |P_L| = 2^r} |E(P_L)| \leq 2|E(C_{v_0})|$. Summing this over all $r \leq \log n$, we find the claimed runtime. □

B Existence of Disjoint Matching

Fix n, $m \leq n^2/4$ and $\epsilon \geq \sqrt{n/m}$. Let $G_1 = (L_1 \cup R_1, E_1)$ consist of $\epsilon^2 n/2$ disjoint copies $B^{(k)}$ of the complete bipartite graph on $1/\epsilon^2$ left and right nodes, containing $2\epsilon^2 m/n$ parallel copies of every edge. In this way, G_1 has n nodes, m edges and is $2m/n$-regular. We index the i-th left and j-th right node of $B^{(k)}$ as $l_i^{(k)}$ resp. $r_j^{(k)}$. Let $G_2 = (L_2 \cup R_2, E_2)$ be the complete bipartite graph on $2m/n$ left nodes and $n/2$ right nodes. In this appendix we prove that every edge in G_1 can be matched to a unique edge in G_2 such that

1. all edges leaving a left node $l \in L_1$ are matched to edges in G_2 with distinct left ends,
2. all edges leaving a right node $r \in R_1$ are matched to edges in G_2 with distinct right ends.

We do this by considering maximum bipartite matchings in G_2 of the form

$$M_j = \{(i, i+j) \mid i \in [2m/n]\} \subset E_2, \qquad 0 \leq j < n/2,$$

where the sum is modulo $n/2$. These matchings form a partition of the edges set E_2. We interpret every M_j as a set of $2m/n$ ordered edges. Now for every node $l_i^{(k)} \in L_1$ we will match the $2m/n$ (lexicographically ordered) outgoing edges $E(l_i^{(k)})$ from that node to the (lexicographically ordered) edges in some M_j. This ensures that all edges leaving $l_i^{(k)}$ are matched to edges in G_2 whose left ends are distinct, so that condition 1. is satisfied. In order to satisfy condition 2., we specify the matching as follows:

$$E(l_i^{(k)}) \Leftrightarrow M_{k-1+(i-1)\epsilon^2 n/2}, \qquad \forall k \in [\epsilon^2 n/2],\ i \in [1/\epsilon^2].$$

Indeed, one can check that the $2m/n$ edges matched to for instance the incoming edges of node $r_1^{(1)}$ are described by

$$(\alpha, \beta), \quad \text{with } \alpha \in [1, 2\epsilon^2 m/n] \text{ and } \beta = \ell \frac{\epsilon^2 n}{2} + \alpha,\ 0 \leq \ell < 1/\epsilon^2.$$

Since we assumed that $m \leq n^2/4$, and hence $2m/n \leq n/2$, the right ends of these edges are indeed disjoint. The same reasoning applies to all other nodes $r_j^{(k)}$. We illustrate this matching in Figure 2 below.

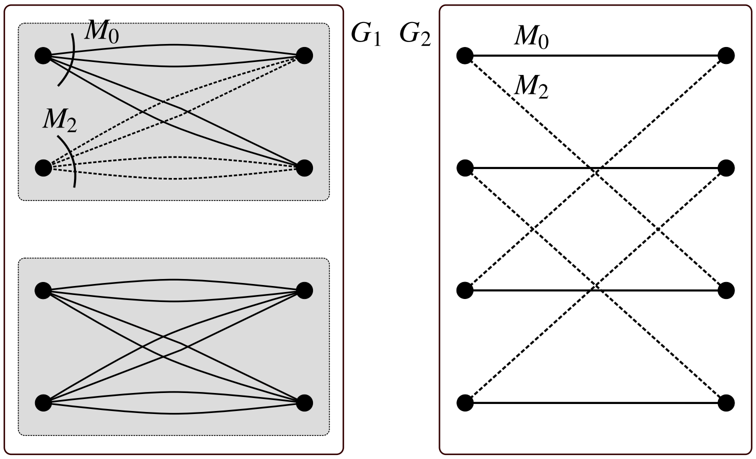

Figure 2: Matching edges between G_1 and G_2 for $n = 8$, $m = 16$ and $\epsilon = 1/\sqrt{2}$. The left ends of matchings M_0 are distinct, ensuring condition 1. for node $l_1^{(1)}$. Similarly, the right ends of all edges matched to $r_1^{(1)}$ are distinct, ensuring condition 2. for this node. We note that in general G_2 is a bipartite graph between $2m/n$ left nodes and $n/2$ right nodes, with $2m/n \leq n/2$.