Quantum Speedup for Graph Sparsification, Cut Approximation and Laplacian Solving

Simon Apers
CWI, Amsterdam, the Netherlands
Inria, Paris, France
ULB, Brussels, Belgium
smgapers@gmail.com

Ronald de Wolf
QuSoft, CWI and University of Amsterdam
Amsterdam, the Netherlands
rdewolf@cwi.nl

***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 demonstrate a polynomial quantum speedup for spectral sparsification and many of its applications. In particular, we give a quantum algorithm that, given a weighted graph with n nodes and m edges, outputs a classical description of an ϵ-spectral sparsifier in sublinear 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). Our algorithm implies a quantum speedup for solving Laplacian systems and for approximating a range of cut problems such as min cut and sparsest cut.**

***Index Terms*—Quantum computing; Quantum algorithms; Graph theory**

I. Introduction and Summary

A. 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 [44] and later Benczúr and Karger [18]. 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 [44], MIN st-CUT [66], [47], [62], SPARSEST CUT and BALANCED SEPARATOR [11], [66]. We refer the interested reader to [63], [67] for surveys on the many applications of cut approximation.

In their breakthrough work on Laplacian solvers, Spielman and Teng [71] 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" [73]. Some examples among these are faster algorithms for learning [80], [79], computer vision and image processing [52], spectral clustering [75], [60] and computing random walk properties [27]. The sparsification results of Spielman and Teng were later refined most notably by Spielman and Srivastava [70] and Batson, Spielman and Srivastava [14]. In [14], 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 [58].

B. Main Result and Applications

We give a *quantum* algorithm for sparsification:

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

The algorithm outputs an explicit classical description, in the form of the list of $\widetilde{O}(n/\epsilon^2)$ edges of the sparsifier together with their new weights. Note the assumption $\epsilon \geq \sqrt{n/m}$. This is because sparsification is only useful when the number of edges of the sparsifier (roughly n/ϵ^2) is at most the number of edges m of the original graph G. Note also that $\widetilde{O}(\sqrt{mn}/\epsilon) \in \widetilde{O}(m)$ whenever $\epsilon \geq \sqrt{n/m}$, and hence our quantum algorithm provides a speedup over classical algorithms, whose $\widetilde{O}(m)$ runtime can be shown to be optimal.[1] For dense graphs, where $m \in \Omega(n^2)$, this improves the time complexity from $\widetilde{O}(n^2)$ classically to $\widetilde{O}(n^{3/2})$ quantumly.

Our algorithm assumes coherent access to the input graph in the form of quantum queries to the adjacency lists. It also assumes a QRAM (coherent RAM) memory of $\widetilde{O}(\sqrt{mn}/\epsilon)$ classical bits to which it can do classical writes, and whose bits it can query in superposition. It uses just $O(\log n)$ "actual" qubits. The "time" (complexity) in the above theorem measures the number of elementary gates, input queries, and QRAM writes and queries. See Section II for more details.

The algorithm builds on a range of quantum and classical results, the most important of which are classical sparsification algorithms by Spielman and Srivastava [70] and Koutis and Xu [51], a spanner algorithm by Thorup and Zwick [74], a quantum algorithm for single-source shortest-path trees by Dürr, Heiligman, Høyer and Mhalla [32] and an efficient k-independent hash function by Christiani, Pagh and Thorup [25].

We prove a matching lower bound, showing that the runtime of our quantum algorithm is optimal up to polylog-factors. In fact, even outputting a weaker *cut* sparsifier requires the same number of queries.

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

[1]Because there is an $\Omega(m)$ query lower bound for deciding whether a graph is connected [33, Theorem 4.9, $k=1$], we have the same linear lower bound for finding a cut sparsifier for a given graph, as well as for applications like approximating MIN CUT.

In this extended abstract we focus on giving the details of our algorithm. However, in the full version of this paper [9] we show that our algorithm also provides a direct speedup for many of the aforementioned applications. In Table I 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	$\widetilde{O}(m)$ [10]	$\widetilde{O}(\sqrt{mn})$
ϵ-MIN CUT	$\widetilde{O}(m)$ [46]	$\widetilde{O}(\sqrt{mn}/\epsilon)$
ϵ-MIN st-CUT	$\widetilde{O}(m + n/\epsilon^5)$ [62]	$\widetilde{O}(\sqrt{mn}/\epsilon + n/\epsilon^5)$
$O(\sqrt{\log n})$-SPAR.CUT/BAL.SEP.	$\widetilde{O}(m + n^{1+\delta})$ [65]	$\widetilde{O}(\sqrt{mn} + n^{1+\delta})$

TABLE I
TIME COMPLEXITY FOR CUT APPROXIMATION. QUANTUM BOUNDS FOLLOW FROM COMBINING OUR QUANTUM SPARSIFICATION ALGORITHM WITH THE BEST CLASSICAL ALGORITHMS. 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. *Fix n, m and $\epsilon \geq \sqrt{n/m}$. There exists a quantum algorithm that, given adjacency-list access to a weighted and undirected n-vertex graph G with m edges and Laplacian L, outputs with high probability an approximate solution $\tilde{x} \in \mathbb{R}^n$ to the linear system $Lx = b$ such that $\|\tilde{x} - x\|_L \leq \epsilon \|x\|_L$ in time $\widetilde{O}(\sqrt{mn}/\epsilon)$.*

This improves over the runtime $\widetilde{O}(m \log(1/\epsilon))$ of classical solvers [72] in terms of m. In contrast to the well-known HHL algorithm [38], our algorithm outputs an explicit classical description of $\tilde{x}$ (i.e., a vector of n real entries), not an n-dimensional quantum state. $\|v\|_L$ 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. Similar to the classical case, we also get a quantum speedup for the more general class of *symmetric, weakly diagonally-dominant* (SDD) linear systems.

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 $\widetilde{O}(1)$, for approximating the random walk cover time, and for 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 there. We summarize our speedups in Table II, and discuss prior work on quantum algorithms for some of these problems in Section I-E.

	Classical	Quantum (this work)
ϵ-Laplacian/SDD solver	$\widetilde{O}(m)$ [72]	$\widetilde{O}(\sqrt{mn}/\epsilon)$
ϵ-eff. resistance (single)	$\widetilde{O}(m)$	$\widetilde{O}(\sqrt{mn}/\epsilon)$
ϵ-eff. resistances (all)	$\widetilde{O}(m+n/\epsilon^4)$ [70]	$\widetilde{O}(\sqrt{mn}/\epsilon+n/\epsilon^4)$
$O(1)$-cover time	$\widetilde{O}(m)$ [30]	$\widetilde{O}(\sqrt{mn})$
k bottom eigenvalues	$\widetilde{O}(m+kn/\epsilon^2)$	$\widetilde{O}(\sqrt{mn}/\epsilon+kn/\epsilon^2)$
spectral clust.	$\widetilde{O}(m+n\operatorname{poly}k)$	$\widetilde{O}(\sqrt{mn}+n\operatorname{poly}k)$

TABLE II
TIME COMPLEXITY OF LAPLACIAN SOLVING AND APPLICATIONS. CLASSICAL BOUNDS WITHOUT REFERENCE ARE FROM [72]. QUANTUM BOUNDS COMBINE OUR QUANTUM SPARSIFICATION ALGORITHM WITH THE BEST CLASSICAL ALGORITHM.

C. *Quantum Algorithm*

Our quantum sparsification algorithm starts from the iterative sparsification algorithm by Koutis and Xu [53]. 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 gives 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 [74] with the shortest-paths quantum algorithm by Dürr, Heiligman, Høyer and Mhalla [32]. 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.

Theorem 4. *Fix n, m. There exists a quantum algorithm that, given adjacency-list access to a weighted and undirected n-vertex graph G with m edges, 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 write down the "intermediate" sparsifiers: after a constant number of iterations these still have $\Omega(m)$ edges, while we aim for runtime $\sqrt{mn}$. We overcome this using two observations, which allow us to describe the intermediate graphs only implicitly.

First we show that *if* we were given query access to a uniformly random 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 random 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 [16]. 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 [25], 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 string:

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

Combining these observations fixes the issue of having to store intermediate graphs, and speeds up the Koutis-Xu algorithm runtime to $\widetilde{O}(\sqrt{mn}/\epsilon^2)$ quantum time.

We further improve the runtime down to $\widetilde{O}(\sqrt{mn}/\epsilon)$ by combining with the sparsification toolbox of Spielman and Srivastava [70]. 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 is also 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 $\widetilde{O}(\sqrt{mn}/\epsilon)$, using Grover's algorithm. This idea of using a "poor" spectral approximation to compute sampling probabilities to obtain a better spectral approximation is also used in [56], [28].

D. Matching Lower Bound

In the full version of this paper [9] 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 [14]). If we can appropriately "hide" these edges among the m edges of G, 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 [7]. 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 [17], prompted by our question to them.

E. Prior Work

We are not aware of prior work on quantum speedups for graph sparsification. In a very different line of work though, sparsification has been studied with the goal of sparsifying Hamiltonian matrices, which are used to describe many-body quantum systems. Aharonov and Zhou [1] 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 [39] considered the weaker notion of sparsifying the Hamiltonian matrix, and suggested that sparsification could indeed help in Hamiltonian simulation.

Research on quantum algorithms for cut approximation is also limited. There is recent work by Hamoudi, Rebentrost, Rosmanis and Santha [37] 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 [12]. Other recent work by Brandão, Kueng and Stilck França [20] 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 [38], which was later refined in work by Ambainis [5] and Childs, Kothari and Somma [24]. 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 [72, Lemma 6.1]). Crucially, they only output a quantum state that encodes the solution, rather than an explicit description as we do.

Quantum speedups for the problems of estimating effective resistances and spectral gaps have also been studied in other work. Very recently and independently from our work, Piddock [64] constructed a quantum walk algorithm that ϵ-approximates the effective resistance $R_{s,t}$ using $\widetilde{O}(\sqrt{mR_{s,t}}/\epsilon^2) \in \widetilde{O}(\sqrt{mn}/\epsilon^2)$ quantum walk steps. He also argued how this could possibly be further improved to $\widetilde{O}(\sqrt{mR_{s,t}}/\epsilon) \in \widetilde{O}(\sqrt{mn}/\epsilon)$. While the quantum walk model is different from our model, this tentative bound would agree with our runtime. In addition, however, we can effectively approximate *all* effective resistances simultaneously in the graph in the same complexity. The problem was also studied in slightly different settings in [76], [23], [40]. 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 [41]. 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 this to $\widetilde{O}(\sqrt{mn}/\epsilon) \in \widetilde{O}(n^{3/2}/\epsilon)$.

We also mention some past and concurrent work on quantum speedups for clustering. One paper by Daskin [29] describes a quantum algorithm for spectral clustering but no direct speedup is found with respect to classical algorithms. Concurrent to our work is a paper by Kerenidis and Landman [49] which describes a quantum algorithm for outputting the centroids of a k-means spectral clustering. In contrast to our work, they start from quantum access to a data set, which they then use to query an associated Laplacian. They find a quantum speedup under certain assumptions (e.g., that the input data is appropriately clustered), and as such is incomparable to our quantum algorithm. Less directly related, there exists a number of papers [3], [57], [77], [50] 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 clustering algorithm.

Finally we mention classical work on sublinear algorithms for Laplacian solving and sparsification. The work by Andoni, Krauthgamer and Pogrow [8] describes a sublinear algorithm for Laplacian solving aimed at 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 [55], who proposes a sublinear algorithm for spectral sparsification of unweighted graphs. He bypasses the $\Omega(m)$ lower bound by allowing a weaker, additive error in the approximation. As such this work is incomparable to ours.

F. Open Questions

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

• We prove a matching $\widetilde{\Omega}(\sqrt{mn}/\epsilon)$ lower bound on the quantum query complexity of spectral sparsification. Can we extend this to a tight lower bound for any of the resulting applications, like ϵ-approximating the min cut or effective resistance? Since these reduce to constructing an ϵ-spectral sparsifier, this would yield a stronger lower bound.

• Work on Laplacian solvers has also led to progress on the long-standing question of computing maximum flows in graphs [26], [66], [47], [62], ultimately leading to classical algorithms with runtime $\widetilde{O}(m)$ that approximate max flows. Since the naive description of such a flow already requires size $\Omega(m)$, this seems optimal, even for quantum algorithms. 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* max flow was already proposed by Ambainis and Špalek [6].

• At first sight, sparsifiers can only yield *approximate* solutions to cut problems. However, for the case of MIN CUT, Karger [45], [46] 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. A related open question, asked by Lee, Santha and Zhang [54] is the quantum complexity of exact MIN CUT w.r.t. *cut queries* (which, given a set S of vertices, return the induced cut value).

• Spectral sparsification of graphs and Laplacians has been extended in different directions such as sparsification of hypergraphs [69], [13], sparsification of sums of positive semi-definite matrices [69], [68], sparsification in a streaming setting [48], [42]. It is also closely related to concepts such as spectral sketching [7] and linear data regression using leverage scores [31]. It seems likely that we can also find quantum speedups for these related problems. Similarly we might hope to solve "quantum" tasks like sparsifying density operators or POVMs.

II. Preliminaries

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

A. Computational Model and Quantum Algorithms

We assume as our computational model a quantum-accessible classical control system that

1) can run quantum subroutines on at most $O(\log N)$ qubits, with N the problem instance size;
2) can make quantum queries to the input; and
3) has access to a quantum-read/classical-write RAM (QRAM)[2] of $\mathrm{poly}(N)$ classical bits, where a single QRAM operation corresponds to either classically writing a bit to the QRAM, or making a quantum query (a read operation) to bits stored in QRAM, possibly in superposition.

In this model, an algorithm has *time complexity* T if it uses at most T elementary classical and quantum gates, quantum queries to the input, and QRAM operations. The *query* complexity of an algorithm only measures the number of queries to the input. If we only care about query complexity, the assumption of having QRAM may be dropped at the expense of a polynomial increase in the number of gates.

An important quantum subroutine in our work is Grover's algorithm [36] 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*

[2]Another name for this type of memory is "coherent RAM." We feel a QRAM containing a classical k-bit string z and allowing efficient queries of the form $|i,b\rangle \mapsto |i, b \oplus z_i\rangle$ (where $i \in [k]$ and $b \in \{0,1\}$) is a reasonable generalization of classical RAM: if one believes in classical RAM and in quantum superposition, then QRAM is quite natural. Like a classical RAM, the physical hardware of such a QRAM necessarily requires size at least proportional to k because it contains k bits of information, but answering a query would have a cost proportional to $\log k$, even when querying multiple stored bits in superposition. This could be realized for instance by laying out the k bits as the leaves of a binary tree of depth $\log k$; the $\log k$ bits of the binary representation of an address $i \in [k]$ would chart a path from the root to the addressed bit z_i, allowing for efficient lookup of the addressed bit. Note that running this on a superposition of different addresses i involves going down different paths in superposition, but still only uses a superposition of $O(\log k)$ qubits.

Assuming such QRAM is very common in quantum algorithms for graph problems. It should be noted, though, that the notion of QRAM is a bit controversial, (1) because the term is sometimes used in the literature for a different and stronger kind of memory (allowing for efficient conversion of a classically-stored unit vector of k numbers into the corresponding state of $\log k$ qubits), and (2) since implementing it on noisy hardware might require $O(k)$ work to error-correct a quantum query, rather than $O(\log k)$ work.

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.

We also use an algorithm for finding shortest-path trees by Dürr, Heiligman, Høyer and Mhalla [32], but the quantum parts in that algorithm reduce to Grover search.

B. 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. [32], [35]. 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.

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

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

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 [4]. Throughout the paper we 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 removing the edges of all previous spanners. Such an r-packing always exists for every r, though once $G - \cup_{i<j} H_i$ has no edges left anymore, the subsequent spanners $H_j, \ldots, H_r$ are empty.

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 $(D_{ii}) = \sum_j w_{ij}$. Equivalently, we can write

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

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 psd.

C. 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 $S \subseteq V$ we have

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

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

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

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 [15], 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 is 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 [43], and further refined by Koutis and Xu [53]. We will build on the latter work which constructs spectral sparsifiers from spanner packings. Their algorithm iteratively invokes the routine described below, which roughly halves the number of edges.

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

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

Theorem 5 ([53, Thm 3.2]). *The graph $H = \textbf{Half Sparsify}(G, \epsilon)$ is, with probability $\geq 1 - 1/n^2$, an ϵ-spectral sparsifier of G with $\leq m/2 + \widetilde{O}(n/\epsilon^2)$ edges.*

Now consider a fixed $\epsilon > 0$. If we iterate $T \in O(\log(m/n))$ times the routine $\textbf{Half Sparsify}(G, \epsilon')$, with $\epsilon' \in O(\epsilon/T)$, then we retrieve with high probability an ϵ-spectral sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges. By [7] this is optimal up to log-factors.

III. Quantum Sparsification Algorithm

Our quantum sparsification 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 V.

As mentioned in the introduction, we cannot simply plug this quantum spanner algorithm in the Koutis and Xu algorithm: after a single iteration this would require to output a graph with up to $m/2$ edges. This is much too costly since we aim at a runtime that scales as $\sqrt{mn}$. To resolve this issue we first assume that we have access to a random 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. Then, we then get rid of the random string. To this end we use efficient k-independent hash functions that allow to simulate queries to a k-wise independent random string. This suffices since by standard results a k-query quantum algorithm cannot distinguish a $2k$-wise independent strings from a uniformly random one.

A. Using a Random String

We first assume access to a family of independent, random 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 random 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 random strings. We show how to do this efficiently in the proof of Theorem 6.

Algorithm 2 $H = $ **Quantum Sparsify**(G, ϵ)

```
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 $w'_e = 0$
6: use repeated Grover to find $H = \{e \in E \mid w'_e > 0\}$
```

Theorem 6. *Given access to independent, uniformly random strings $r_i \in \{0,1\}^m$, $i \in [\log(m/n)]$, algorithm* **Quantum Sparsify**(G, ϵ) *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 follows from Theorem 5: per iteration we "half-sparsify" the 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 [59], the repeated Grover search routine in the final step takes time $\widetilde{O}(\sqrt{mn}/\epsilon)$ for finding n/ϵ^2 edges among m edges.

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

The space complexity of the algorithm is $O(\log n)$ 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 number of classical bits is dominated by the output size.

B. Using k-independent Hash Functions

To get rid of the random strings $\{r_i\}$ we build on an easy consequence of the polynomial method [16]. It seems that this was first used in the proof of [21, Theorem 19], and is stated in e.g. [78, Theorem 3.1].

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

As a consequence, we can replace the uniformly random strings of length m by a k-wise independent 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 [22]. Crucial, 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 [25], who proved the theorem below.

Theorem 7 ([25]). *It is possible to construct in time $\widetilde{O}(k)$ a data structure of size $\widetilde{O}(k)$ that simulates 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 string. We can simulate this algorithm with a quantum algorithm with runtime $\widetilde{O}(q)$ without random string, using $\widetilde{O}(q)$ additional classical bits.*

Hence we can efficiently simulate the random string in Algorithm 2, with at most a polylogarithmic overhead in the runtime. The classical space complexity of the algorithm increases from $\widetilde{O}(n/\epsilon^2)$ to $\widetilde{O}(\sqrt{mn}/\epsilon^2)$. Combining Theorem 6 with Corollary 1 gives:

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 $\widetilde{O}(\sqrt{mn}/\epsilon^2)$. It uses $O(\log n)$ qubits and $\widetilde{O}(\sqrt{mn}/\epsilon^2)$ classical bits.*

IV. Refined Quantum Sparsification

Here we show how to improve the runtime from $\widetilde{O}(\sqrt{mn}/\epsilon^2)$ to $\widetilde{O}(\sqrt{mn}/\epsilon)$, which matches our lower bound up to polylog-factors. The improvement builds on seminal results by Spielman and Srivastava [70]. They first showed that sampling edges with probabilities approximately proportional to their effective resistances results in a spectral sparsifier. Then they showed how Laplacian solvers can efficiently estimate these effective resistances. We use our quantum sparsification algorithm to first construct a "rough" ϵ-sparsifier, for constant ϵ, which we use to approximate 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, 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 [56], [28].

A. 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 [70], and a quantum sampling routine for implementing this sparsification algorithm.

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 ([70]). *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 $\geq 1 - 1/n$, for every pair $s, t \in V$ we have $(1-\epsilon)R_{s,t} \leq \|Z(\chi_s - \chi_t)\|^2 \leq (1+\epsilon)R_{s,t}$.*

This Z is 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 $\widetilde{O}(1/\epsilon^2)$.

2) Spectral Sparsification with Edge Scores: Spielman-Srivastava also showed a spectral sparsifier can be constructed by independently keeping edges with weights roughly proportional to effective resistance.

Theorem 10 ([70]). *Let $2R_e \geq \widetilde{R}_e \geq R_e/2$ for each edge $e \in E$, and $p_e = \min(1, C w_e \widetilde{R}_e \log(n)/\epsilon^2)$ 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 $\sum_e w_e R_e = n - 1$ [19, Theorem 25], this yields the claimed number of edges.[3]

3) Quantum Sampling: Assuming access to an approximate resistance oracle that gives approximations $\widetilde{R}_e$ to R_e, we implement the Spielman-Srivastava sparsification scheme. Classically this requires time $\widetilde{O}(m + \sum_e p_e)$, but quantumly it can be done 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*

[3]Spielman and Srivastava describe a slightly different scheme: sample $\widetilde{O}(n/\epsilon^2)$ i.i.d. edges, with probabilities proportional to effective resistance. This gives the same performance bound [34, Remark 1].

every e independently with probability p_e, in expected time $\widetilde{O}(\sqrt{m(\sum_e p_e)})$.

Proof. By Corollary 1 we can assume access to a random $\widetilde{O}(m)$-bit random string r. From this r we can derive a function $h_r : E \times [0,1] \to \{0,1\}$ s.t. for each e independently $h_r(e, p_e) = 1$ with probability p_e. Combining with a query to the list of probabilities allows to implement an oracle $|e\rangle |0\rangle |0\rangle \mapsto |e\rangle |p_e\rangle |h_r(e,p_e)\rangle$. Let T be the number of $e \in E$ for which $h_r(e,p_e) = 1$. Then $\mathbb{E}[T] = \sum_e p_e$. Use repeated Grover search (Claim 2) to retrieve these edges in expected time $\widetilde{O}(\sqrt{mT})$. $\square$

B. Refined Quantum Sparsification

Now we combine the Spielman-Srivastava toolbox, the quantum sampling routine and our quantum sparsification algorithm from the last section.

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

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

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, the total success probability is $\geq (1-1/n)^3 = 1 - O(1/n)$.

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 $\sum_e w_e R^G_e = n-1$ [19, 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. $\square$

V. Quantum Algorithm for Finding 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 II 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 holds that

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

where $\delta_G(u,v) = \min_{u-v \text{ path } P} \sum_{e\in P} 1/w_e$. 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. [4] 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 [61] or the very recent survey by Ahmed et al. [2]. There exists a range of classical algorithms for constructing spanners. We will make use of one by Thorup and Zwick [74], 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 using the quantum algorithm of Dürr, Heiligman, Høyer and Mhalla [32] for growing a shortest-path tree in time $\widetilde{O}(\sqrt{mn})$.

A. Thorup-Zwick Algorithm

The Thorup-Zwick spanner algorithm [74] uses *shortest-path trees* (SPTs). An SPT $T(v)$ 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 the distance in the original graph G. Algorithm 4 randomly partitions the node set into k layers $\{A_i\}$, which are increasingly sparsified. The nodes in these layers are “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)

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

Theorem 12 (from the analysis in [74])**.**

- *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 $k = 1/2 + \log n$ yields a $2 \log n$-spanner with an expected number of edges $O(n \log n)$.

B. Quantum Spanner Algorithm

We can use a quantum algorithm from Dürr, Heiligman, Høyer and Mhalla [32] 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 the full version [9] 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.*

To speed up the spanner construction, 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, using Cauchy-Schwarz the total runtime becomes $\widetilde{O}(\sum_v \sqrt{|C(v)||E(C(v))|}) \in \widetilde{O}(\sqrt{\sum_v |C(v)|}\sqrt{\sum_v |E(C(v))|})$. By Theorem 12 we know that $\mathbb{E}(\sum_v |C(v)|) \in O(kn^{1+1/k})$ and that $\mathbb{E}(\sum_v |E(C(v))|) \in O(kmn^{1/k})$. By Markov's inequality, 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 [32] 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.

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 also benefited from discussions with Joran van Apeldoorn, André Chailloux, Shantanav Chakraborty, András Gilyén, Michael Kapralov, Robin Kothari, Anthony Leverrier, Christian Majenz, Christian Schaffner, Luca Trevisan and Luca Zanetti. We also thank the anonymous FOCS referees for helpful comments. SA was at the CWI-Inria International Lab when most of this work was done. RdW was partially supported by the

Dutch Research Council (NWO) through Gravitation-grant Quantum Software Consortium 024.003.037, and QuantERA project QuantAlgo 680-91-034.

References

[1] D. Aharonov and L. Zhou, "Hamiltonian sparsification and gap-simulation," in *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, 2019, pp. 2:1–2:21, arXiv: 1804.11084

[2] R. Ahmed, G. Bodwin, F. D. Sahneh, K. Hamm, M. J. Jebelli, S. Kobourov, and R. Spence, "Graph spanners: A tutorial review," 2019, arXiv: 1909.03152

[3] E. Aïmeur, G. Brassard, and S. Gambs, "Quantum clustering algorithms," in *Proceedings of the 24th International Conference on Machine Learning (ICML)*. ACM, 2007, pp. 1–8, doi: 10.1145/1273496.1273497

[4] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares, "On sparse spanners of weighted graphs," *Discrete & Computational Geometry*, vol. 9, no. 1, pp. 81–100, 1993.

[5] A. 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)*, vol. 14. LIPIcs, 2012, pp. 636–647.

[6] A. Ambainis and R. Špalek, "Quantum algorithms for matching and network flows," in *Proceedings of the 23rd Symposium on Theoretical Aspects of Computer Science (STACS)*. Springer, 2006, pp. 172–183. [Online]. Available: https://doi.org/10.1007/11672142_13

[7] A. Andoni, J. Chen, R. Krauthgamer, B. Qin, D. P. Woodruff, and Q. Zhang, "On sketching quadratic forms," in *Proceedings of the 2016 Innovations in Theoretical Computer Science Conference (ITCS)*. ACM, 2016, pp. 311–319.

[8] A. Andoni, R. Krauthgamer, and Y. Pogrow, "On solving linear systems in sublinear time," in *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*. ACM, 2019, pp. 3:1–3:19.

[9] S. Apers and R. de Wolf, "Quantum speedup for graph sparsification,cut approximation and Laplacian solving," 2020, arXiv: 1911.07306

[10] S. Arora and S. Kale, "A combinatorial, primal-dual approach to semidefinite programs," *Journal of the ACM*, vol. 63, no. 2, 2016.

[11] S. Arora, S. Rao, and U. Vazirani, "Expander flows, geometric embeddings and graph partitioning," *Journal of the ACM*, vol. 56, no. 2, p. 5, 2009.

[12] B. Axelrod, Y. P. Liu, and A. Sidford, "Near-optimal approximate discrete and continuous submodular function minimization," in *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2020, pp. 837–853.

[13] N. Bansal, O. Svensson, and L. Trevisan, "New notions and constructions of sparsification for graphs and hypergraphs," in *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2019, pp. 910–928.

[14] J. Batson, D. A. Spielman, and N. Srivastava, "Twice-Ramanujan sparsifiers," *SIAM Journal on Computing*, vol. 41, no. 6, pp. 1704–1721, 2012.

[15] J. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng, "Spectral sparsification of graphs: theory and algorithms," *Communications of the ACM*, vol. 56, no. 8, pp. 87–94, 2013.

[16] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, "Quantum lower bounds by polynomials," *Journal of the ACM*, vol. 48, no. 4, pp. 778–797, 2001.

[17] A. Belovs and T. Lee, "The quantum query complexity of composition with a relation," 2020, arXiv: 2004.06439

[18] A. A. Benczúr and D. R. Karger, "Approximating $s-t$ minimum cuts in $\widetilde{O}(n^2)$ time," in *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC)*, vol. 96. ACM, 1996, pp. 47–55.

[19] B. Bollobás, *Modern graph theory*. Springer Science & Business Media, 2013, vol. 184.

[20] F. G. S. L. Brandão, R. Kueng, and D. Stilck França, "Faster quantum and classical SDP approximations for quadratic binary optimization," 2019, arXiv: 1909.04613

[21] H. Buhrman, L. Fortnow, I. Newman, and H. Röhrig, "Quantum property testing," *SIAM Journal on Computing*, vol. 37, no. 5, pp. 1387–1400, 2008.

[22] L. J. Carter and M. N. Wegman, "Universal classes of hash functions," *Journal of computer and system sciences*, vol. 18, no. 2, pp. 143–154, 1979.

[23] S. Chakraborty, A. Gilyén, and S. 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)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, pp. 33:1–33:14.

[24] A. M. Childs, R. Kothari, and R. D. Somma, "Quantum algorithm for systems of linear equations with exponentially improved dependence on precision," *SIAM Journal on Computing*, vol. 46, no. 6, pp. 1920–1950, 2017.

[25] T. Christiani, R. Pagh, and M. Thorup, "From independence to expansion and back again," in *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC)*. ACM, 2015, pp. 813–820.

[26] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. 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)*. ACM, 2011, pp. 273–282.

[27] M. B. Cohen, J. A. Kelner, J. Peebles, R. Peng, A. Sidford, and A. 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)*. IEEE, 2016, pp. 583–592.

[28] M. B. Cohen, Y. T. Lee, C. M. Musco, C. P. Musco, R. Peng, and A. Sidford, "Uniform sampling for matrix approximation," in *Proceedings of the 6th Innovations in Theoretical Computer Science Conference (ITCS)*. ACM, 2015, pp. 181–190.

[29] A. Daskin, "Quantum spectral clustering through a biased phase estimation algorithm," 2017, arXiv: 1703.05568

[30] J. Ding, J. R. Lee, and Y. Peres, "Cover times, blanket times, and majorizing measures," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 2011, pp. 61–70.

[31] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, "Sampling algorithms for l_2 regression and applications," in *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 2006, pp. 1127–1136.

[32] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla, "Quantum query complexity of some graph problems," *SIAM Journal on Computing*, vol. 35, no. 6, pp. 1310–1328, 2006.

[33] T. Eden and W. Rosenbaum, "Lower bounds for approximating graph parameters via communication complexity," in *Proceedings of APPROX-RANDOM*, 2018, pp. 11:1–11:18.

[34] W.-S. Fung, R. Hariharan, N. J. A. Harvey, and D. Panigrahi, "A general framework for graph sparsification," *SIAM Journal on Computing*, vol. 48, no. 4, pp. 1196–1223, 2019.

[35] O. Goldreich and D. Ron, "Property testing in bounded degree graphs," *Algorithmica*, vol. 32, no. 2, pp. 302–343, 2002.

[36] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC)*. ACM, 1996, pp. 212–219.

[37] Y. Hamoudi, P. Rebentrost, A. Rosmanis, and M. Santha, "Quantum and classical algorithms for approximate submodular function minimization," *Quantum Information and Computation*, vol. 19, no. 15-16, pp. 1325–1349, 2019.

[38] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical Review Letters*, vol. 103, no. 15, p. 150502, 2009.

[39] S. Herbert and S. Subramanian, "Spectral sparsification of matrix

inputs as a preprocessing step for quantum algorithms," 2019, arXiv: 1910.02861

[40] T. Ito and S. Jeffery, "Approximate span programs," *Algorithmica*, vol. 81, no. 6, pp. 2158–2195, 2019.

[41] M. Jarret, S. Jeffery, S. Kimmel, and A. Piedrafita, "Quantum algorithms for connectivity and related problems," in *Proceedings of the 26th European Symposium on Algorithms (ESA)*. Springer, 2018, pp. 49:1–49:13.

[42] M. Kapralov, Y. T. Lee, C. M. Musco, C. P. Musco, and A. Sidford, "Single pass spectral sparsification in dynamic streams," *SIAM Journal on Computing*, vol. 46, no. 1, pp. 456–477, 2017.

[43] M. Kapralov and R. Panigrahy, "Spectral sparsification via random spanners," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*. ACM, 2012, pp. 393–398.

[44] D. R. Karger, "Using randomized sparsification to approximate minimum cuts," in *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, vol. 94, 1994, pp. 424–432.

[45] ——, "Random sampling in cut, flow, and network design problems," *Mathematics of Operations Research*, vol. 24, no. 2, pp. 383–413, 1999.

[46] ——, "Minimum cuts in near-linear time," *Journal of the ACM*, vol. 47, no. 1, pp. 46–76, 2000.

[47] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. 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)*. SIAM, 2014, pp. 217–226.

[48] J. A. Kelner and A. Levin, "Spectral sparsification in the semi-streaming setting," *Theory of Computing Systems*, vol. 53, no. 2, pp. 243–262, 2013.

[49] I. Kerenidis and J. Landman, "Quantum spectral clustering," 2020, arXiv: 2007.00280

[50] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash, "q-means: A quantum algorithm for unsupervised machine learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 4136–4146.

[51] I. Koutis, G. L. Miller, and R. Peng, "Approaching optimality for solving SDD linear systems," *SIAM Journal on Computing*, vol. 43, no. 1, pp. 337–354, 2014.

[52] I. Koutis, G. L. Miller, and D. Tolliver, "Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing," *Computer Vision and Image Understanding*, vol. 115, no. 12, pp. 1638–1646, 2011.

[53] I. Koutis and S. C. Xu, "Simple parallel and distributed algorithms for spectral graph sparsification," *ACM Transactions on Parallel Computing (TOPC)*, vol. 3, no. 2, pp. 1–14, 2016.

[54] T. Lee, M. Santha, and S. Zhang, "Quantum algorithms for graph problems with cut queries," 2020, arXiv: 1709.04262

[55] Y. T. Lee, "Probabilistic spectral sparsification in sublinear time," 2013, arXiv: 1401.0085

[56] M. Li, G. L. Miller, and R. Peng, "Iterative row sampling," in *Proceedings of the 54th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2013, pp. 127–136.

[57] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," 2013, arXiv: 1307.0411

[58] A. W. Marcus, D. A. Spielman, and N. Srivastava, "Interlacing families II: Mixed characteristic polynomials and the Kadison-Singer problem," *Annals of Mathematics*, pp. 327–350, 2015.

[59] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*. Cambridge University Press, 2002.

[60] L. Orecchia, S. Sachdeva, and N. 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)*. ACM, 2012, pp. 1141–1160.

[61] D. Peleg, "Distributed computing," *SIAM Monographs on discrete mathematics and applications*, vol. 5, 2000.

[62] R. Peng, "Approximate undirected maximum flows in $O(m \operatorname{polylog}(n))$ time," in *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2016, pp. 1862–1867.

[63] J.-C. Picard and M. Queyranne, "Selected applications of minimum cuts in networks," *INFOR: Information Systems and Operational Research*, vol. 20, no. 4, pp. 394–422, 1982.

[64] S. Piddock, "Quantum walk search algorithms and effective resistance," 2019, arXiv: 1912.04196

[65] J. 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)*. IEEE, 2009, pp. 363–372.

[66] ——, "Nearly maximum flows in nearly linear time," in *Proceedings of the 54th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2013, pp. 263–269.

[67] D. B. Shmoys, "Approximation algorithms for np-hard problems." PWS Publishing Co., 1997, ch. Cut Problems and Their Application to Divide-and-conquer, pp. 192–235.

[68] M. K. Silva, N. J. A. Harvey, and C. M. Sato, "Sparse sums of positive semidefinite matrices," *ACM Transactions on Algorithms (TALG)*, vol. 12, no. 1, p. 9, 2016.

[69] T. Soma and Y. Yoshida, "Spectral sparsification of hypergraphs," in *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2019, pp. 2570–2581.

[70] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1913–1926, 2011.

[71] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011.

[72] ——, "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 3, pp. 835–885, 2014.

[73] S.-H. Teng, "The Laplacian paradigm: Emerging algorithms for massive graphs," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2010, pp. 2–14.

[74] M. Thorup and U. Zwick, "Approximate distance oracles," *Journal of the ACM*, vol. 52, no. 1, pp. 1–24, 2005.

[75] N. K. Vishnoi, "Lx= b," *Foundations and Trends in Theoretical Computer Science*, vol. 8, no. 1–2, pp. 1–141, 2013.

[76] G. Wang, "Efficient quantum algorithms for analyzing large sparse electrical networks," *Quantum Information and Computation*, vol. 17, no. 11-12, pp. 987–1026, 2017.

[77] N. Wiebe, A. Kapoor, and K. Svore, "Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning," *Quantum Information and Computation*, vol. 15, no. 3&4, pp. 316–356, 2015.

[78] M. Zhandry, "Secure identity-based encryption in the quantum random oracle model," *International Journal of Quantum Information*, vol. 13, no. 04, p. 1550014, 2015.

[79] D. Zhou, J. Huang, and B. Schölkopf, "Learning from labeled and unlabeled data on a directed graph," in *Proceedings of the 22nd International Conference on Machine Learning (ICML)*. ACM, 2005, pp. 1036–1043.

[80] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003, pp. 912–919.