(Sub)Exponential advantage of adiabatic quantum computation with no sign problem*

Matthew B. Hastings András Gilyén Umesh Vazirani

We demonstrate the possibility of (sub)exponential quantum speedup via a quantum algorithm that follows an adiabatic path of a gapped sparse Hamiltonian with no sign problem. This is in sharp contrast with frustration-free stoquastic Hamiltonians, where no such speedup is possible as shown by Bravyi and Terhal [BT10]. The Hamiltonian that exhibits this speed-up comes from the adjacency matrix of an undirected graph, and we can view the adiabatic evolution as an efficient $\mathcal{O}(\mathrm{poly}(n))$-time quantum algorithm for finding a specific "EXIT" vertex in the graph given the "ENTRANCE" vertex. On the other hand we show that if the graph is given via an adjacency-list oracle, there is no classical algorithm that finds the "EXIT" with probability greater than $\exp(-n^\delta)$ using at most $\exp(n^\delta)$ queries for $\delta = \frac{1}{5} - o(1)$. Our construction of the graph is somewhat similar to the "welded-trees" construction of Childs et al. [CCD+03], but uses additional ideas for simultaneously achieving a spectral gap and a short adiabatic path.

Adiabatic quantum computing [FGGS00] is an interesting model of computation that is formulated directly in terms of Hamiltonians, the quantum analog of constraint satisfaction problems (CSPs). The computation starts in the known ground state of an initial Hamiltonian, and slowly (adiabatically) transforms the acting Hamiltonian into a final Hamiltonian whose ground state encapsulates the answer to the computational problem in question. The final state of the computation is guaranteed, by the quantum adiabatic theorem, to have high overlap with the desired ground state, as long as the running time of the adiabatic evolution is polynomially large in the inverse of the smallest spectral gap of any Hamiltonian along the adiabatic path [AR04]. This model has been intensely studied, not only because of its inherent interest, but also because it is the zero-temperature limit of quantum annealing.

In full generality, adiabatic quantum computing is known to be equivalent to standard circuit-based quantum computing [AvDK+07]. A very interesting question is what is the power of adiabatic quantum computing where all Hamiltonians were "stoquastic," i.e., restricted to not having a sign problem. What this means is that in some basis all off-diagonal terms of H are non-positive. Adiabatic quantum computing with no sign problem includes the most natural case where the final Hamiltonian is diagonal, and represents the objective function to be optimized, and the initial Hamiltonian consists of Pauli X operators acting on each qubit, with ground state the uniform superposition on all the n-bit strings. This question was also motivated by understanding the computational limits of the quantum annealers implemented by the company D-Wave, where all the Hamiltonians were stoquastic.

Bravyi and Terhal [BT10] showed that for frustration-free Hamiltonians without a sign problem, computing the ground state is classically tractable, thereby raising the question of whether this was true for general Hamiltonians without a sign problem. Indeed, a stronger conjecture was that quantum Monte-Carlo, a widely used heuristic in computational condensed matter physics, already provided a technique for an efficient classical simulation. This latter possibility was ruled out by a result of Hastings and Freedman [HF13], who showed the existence of topological obstructions to the convergence of quantum Monte Carlo on such problems.

The question of classical tractability for general Hamiltonians with no sign problem was still open. In this submission, we show that there is a (sub)exponential oracle separation, of the form 2^{n^δ} between classical algorithms and adiabatic quantum computation with no sign problem. Moreover, our construction of the Hamiltonian is fairly transparent – it consists of a graph with an ENTRANCE and an EXIT vertex, and the challenge is: given the ENTRANCE vertex and oracle access to the adjacency matrix of the graph, find the EXIT vertex. A simple quantum walk finds the EXIT vertex in polynomial time, and likewise so does a simple adiabatic algorithm which carries out a straight line interpolation between the initial and final Hamiltonian.

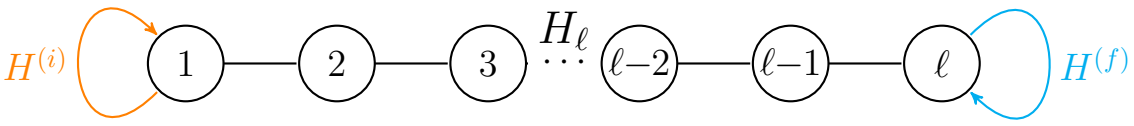
*This submission is based on two recent papers: [Has20, GV20].

Our simple construction highlights the similarities and differences with the well known "welded-trees" graph (see Fig. 1 of the paper) which is the basis of the first known example of exponential speedup by quantum walks [CCD+03]. The welded-trees graph is not suitable for adiabatic computation, since the ground state has exponentially small support on the roots of the two trees (the ENTRANCE and EXIT vertices). To see this, notice that a quantum walk on the welded trees may be viewed as walking on the symmetric subspace of each level of the trees – i.e. it is just a path of length 2depth + 1. This path has uniform edge weights, except at the middle edge, which has $\sqrt{2}$-times bigger weight. This makes the largest eigenvector of the path graph decay exponentially from the middle towards the two ends. We start by the simple observation that equalizing the edge weights in the level graph on symmetric subspaces has the effect of fixing the exponential decay problem. On the other hand, this necessarily makes the underlying graph non-regular. In turn, this enables classical algorithms to detect the structure of the graph [CW20, BDCG+20] and ultimately destroys the lower bound of [CCD+03] that heavily build on the regularity of the graph. In order to restore the classical hardness result we "decorate" the graph by means of attaching a carefully shaped forest to every vertex. Another feature that can be seen very concretely in our simplified construction is the role of the ℓ_2 versus ℓ_1 normalization difference in the behavior of the quantum vs. classical walk.

Main results. The main idea is to start with a graph that the adiabatic algorithm can traverse efficiently, and to hide that graph within a larger graph as follows: attach a number of trees to each vertex of the original graph, so that the attached trees form the bulk of the new graph. Now, the intuition is that the behavior of a quantum walk versus classical walk on the attached trees would be governed by their $\ell_2 \to \ell_2$ (i.e., spectral) norm versus $\ell_1 \to \ell_1$ norm respectively, and the latter is quadratically larger. As a result the attached trees only negligibly affect the ℓ_2-weight distribution of the ground state (and so quantum algorithms only suffer from a minor perturbation), while they dramatically shift the ℓ_1-weight distribution of the ground state away from the original graph. Intuitively speaking this enables the trees to lure away classical random walks from the original graph, so that they get lost in the attached "camouflage trees". Furthermore, by choosing the trees to have a confusing enough shape, one can ensure that there is no classical algorithm that can avoid getting drawn into the "camouflage trees." Therefore, classical algorithms fail to quickly explore the original graph, and in our case this ultimately leads to their inability of efficiently finding the EXIT vertex.

The classical hardness is achieved by constructing hard-to-navigate trees with a fractal-like structure that are built in a recursive manner, via a sequence of so-called "decorations". In order to achieve (sub)exponential hardness, we apply polynomially many rounds of decoration. Intuitively speaking each round of decoration doubles the time that a classical algorithm needs for finding the EXIT.

The basic adiabatic path at the core of our quantum algorithm. We begin with a simple underlying problem of starting at one endpoint of a path on ℓ vertices, and finding the other endpoint of the path. A simple adiabatic algorithm for this problem is specified as follows: Let A_ℓ denote the adjacency matrix of the path $\langle k|A_\ell|k+1\rangle = \langle k+1|A_\ell|k\rangle = 1$, and let the corresponding Hamiltonian be $H_\ell := -A_\ell$, while $H^{(i)} := -|1\rangle\langle 1|$ and $H^{(f)} := -|\ell\rangle\langle \ell|$.



Consider the simple adiabatic path $H_\ell(s)$ that first interpolates between H_i and H_ℓ, then between H_ℓ and H_f, so that $H_\ell(s) := (1+s)H_\ell - sH^{(i)}$ for $s \in [-1, 0]$ and $H_\ell(s) := (1-s)H_\ell + sH^{(f)}$ for $s \in [0, 1]$.

If one moves slowly enough along this adiabatic path [FGGS00, AR04], the quantum evolution maps "ENTRANCE":= $|1\rangle$ – the *initial* ground state of $H^{(i)}$ to "EXIT":= $|\ell\rangle$ – the *final* ground state of $H^{(f)}$, since $H(s)$ has a gap of size $\Omega(\frac{1}{\ell^2})$ for all $s \in [-1, 1]$. Note that if one wishes to use only

a simple "straight" adiabatic path, and stops at $s = 0$, a measurement in the computational basis still reveals the state $|\ell\rangle$ with probability at least $\Omega(\ell^{-3})$ since the ground state of H_ℓ has $\Omega(\ell^{-\frac{3}{2}})$ overlap with $|\ell\rangle$.

Making the task of finding EXIT classically hard. In order to prove classical hardness we will hide the EXIT vertex in a larger graph – the new graph will be chosen to allow the quantum adiabatic algorithm to still be efficient, while making the task of any classical algorithm very difficult. The id's of the vertices will be chosen randomly in order to remove any non-structural hints about the whereabouts of the EXIT vertex, and the graph will be specified by oracle access to its adjacency list, together with the ENTRANCE vertex – one of the two vertices with a self-loop. The task is to find the EXIT vertex – the other vertex with a self-loop attached. The graph will have polynomially bounded maximal vertex degree, so the adiabatic evolution can be efficiently performed by a quantum computer using (time-dependent) sparse Hamiltonian simulation techniques [BCS+20].

In order to make the task of finding EXIT classically hard we "blow-up" the path graph of length ℓ via two main modifications, that we call *obfuscation* and *decoration*.

Definition 1 (Obfuscation of a path of length ℓ)**.** *We replace every vertex that has distance $d \in [k]$ from terminal vertices $\{ENTRANCE, EXIT\}$ by a cluster C of m^{2d} vertices and call these the* funnel *vertices, and replace the other middle vertices (that have distance $d > k$) by a cluster of m^{2k} vertices, and call those the* tunnel *vertices. Then we add edges between clusters C_j and C_{j+1} corresponding to neighbor vertices j and $j+1$ in P_ℓ, so that we build an m^2-ary tree (with the terminal vertices as roots) on the funnel vertices. Between clusters that correspond to vertices with distance $d \geq k$ we add edges along m random matchings. Additionally, in order to preserve spectral properties we add $2m$ self-loops to the ENTRANCE and the EXIT vertices, and an independently chosen random uniform degree-$(4 \cdot m)$ expander graph on each cluster $C_j \colon j \in [\ell] \setminus \{1, \ell\}$, as in Appendix C.*

Note that the graph on the tunnel vertices is $4m$-regular. The decoration construction, described next, will hang m trees from each vertex of the obfuscated graph, each of them being a complete $(5m-1)$-ary tree (by a complete tree we mean a tree for which every node has the same number of children except at the bottom layer, which is at a fixed depth) on its first $\mathrm{poly}(m)$ layers, and then having gradually less children at later layers. The construction is motivated by its effect on the tunnel — it will increase the degree of each tunnel vertices to $5m$. Thus, the resulting graph will still be $5m$-regular on the original tunnel vertices, as well as on the surrounding vertices in the first $\mathrm{poly}(m)$ layers of the added trees. This will make it very difficult for any classical algorithm to distinguishing edges between the tunnel vertices from edges that lead away from the tunnel, thereby making the traversal of the tunnel very slow.

If we would everywhere add a complete $(5m-1)$-ary tree of depth d, then the decoration trees would be easy to detect: after traversing an edge perform a non-backtracking walk of length d, if one arrives at a leaf it means that the traversed edge is hanging a decoration tree. Since we cannot add more than $\exp(\mathrm{poly}(m))$ new vertices, the trees must have a bounded depth. Therefore, in order to circumvent such detection algorithms we should construct trees where the distribution of the lengths before a non-backtracking random walk hits the bottom of a tree looks approximately self-similar, i.e., after going one level deeper in the tree the expected distribution should not change by more than a (sub)exponentially small amount. In order to achieve this, the decoration is carried out in $r = m^\delta$ rounds, giving the attached trees a complex fractal-like structure.

Definition 2 (Decoration)**.** *Let $G = (V, E)$ be a graph. A level-j decoration graph G_j is obtained from G by "decorating" every vertex $v \in V$ by attaching $m^{(1-\delta)}$ new trees via an edge to their root. The attached trees are complete $(5m-(j-1)m^{(1-\delta)}-1)$-ary trees with depth $jm^{(3\delta+o(1))}$. We define $G^{(r)}$ as the r-round decoration of G, which is obtained from G by applying a level-r decoration, then subsequently level-$(r-1)$, level-$(r-2)$, ..., level-1 decorations.*

References

[AvDK+07] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Journal on Computing*, 37(1):166–194, 2007. arXiv: quant-ph/0405098

[AR04] Andris Ambainis and Oded Regev. An elementary proof of the quantum adiabatic theorem. arXiv: quant-ph/0411152, 2004.

[BCS+20] Dominic W. Berry, Andrew M. Childs, Yuan Su, Xin Wang, and Nathan Wiebe. Time-dependent Hamiltonian simulation with L^1-norm scaling. *Quantum*, 4:254, 2020. arXiv: 1906.07115

[BDCG+20] Shalev Ben-David, Andrew M. Childs, András Gilyén, William Kretschmer, Supartha Podder, and Daochen Wang. Symmetries, graph properties, and quantum speedups. In *Proceedings of the 61st IEEE Symposium on Foundations of Computer Science (FOCS)*, 2020. To appear. arXiv: 2006.12760

[BT10] Sergey Bravyi and Barbara Terhal. Complexity of stoquastic frustration-free hamiltonians. *SIAM Journal on Computing*, 39(4):1462–1485, 2010. arXiv: 0806.1746

[CCD+03] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the 35th ACM Symposium on the Theory of Computing (STOC)*, pages 59–68, 2003. arXiv: quant-ph/0209131

[CW20] Andrew M. Childs and Daochen Wang. Can graph properties have exponential quantum speedup? arXiv: 2001.10520, 2020.

[FGGS00] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. arXiv: quant-ph/0001106, 2000.

[GV20] András Gilyén and Umesh Vazirani. (Sub)Exponential advantage of adiabatic quantum computation with no sign problem. arXiv: 2011.09495, 2020.

[Has20] Matthew B. Hastings. The power of adiabatic quantum computation with no sign problem. arXiv: 2005.03791, 2020.

[HF13] Matthew B. Hastings and Michael H. Freedman. Obstructions to classically simulating the quantum adiabatic algorithm. *Quantum Information and Computation*, 13(11&12):1038–1076, 2013. arXiv: 1302.5733