J Syst Sci Complex (2019) 32: 375–452

Quantum Algorithm Design: Techniques and Applications*

SHAO Changpeng · LI Yang · LI Hongbo

DOI: 10.1007/s11424-019-9008-0
Received: 9 October 2018 / Revised: 31 December 2018

Abstract In recent years, rapid developments of quantum computer are witnessed in both the hardware and the algorithm domains, making it necessary to have an updated review of some major techniques and applications in quantum algorithm design.

In this survey as well as tutorial article, the authors first present an overview of the development of quantum algorithms, then investigate five important techniques: Quantum phase estimation, linear combination of unitaries, quantum linear solver, Grover search, and quantum walk, together with their applications in quantum state preparation, quantum machine learning, and quantum search. In the end, the authors collect some open problems influencing the development of future quantum algorithms.

Keywords Quantum algorithm, quantum computation, quantum machine learning, quantum search, quantum walk.

1 Overview: Development of Quantum Algorithms

Quantum computer is based on a computational model obeying the laws of quantum mechanics. Quantum computing is regarded as a highly promising research direction in computer science. Applications of quantum computing range from breaking cryptographic systems to solving real-world problems related to big data, AI, communication, new medicine and material, etc. Although large-scale general-purpose quantum computers are still not in existence, the theory of *quantum algorithms*, i.e., algorithms that run on quantum computer, has been an active research field for over 30 years.

SHAO Changpeng · LI Yang · LI Hongbo (Corresponding author)
Academy of Mathematics and Systems Science, University of Chinese Academy of Sciences, Chinese Academy of Sciences, Beijing 100190, *China.*
Email: shaochangpeng11@mails.ucas.ac.cn; liyang815@mails.ucas.ac.cn; hli@mmrc.iss.ac.cn.
*This research was supported partially by the National Natural Science Foundation of China under Grant No. 11671388, CAS Project QYZDJ-SSW-SYS022, and GF S&T Innovation Special Zone Project.
◇ *This paper was recommended for publication by Editor-in-Chief GAO Xiao-Shan.*

1.1 Early History

In 1980, Benioff[1] first pointed out the possibility of constructing a computer based on quantum mechanics. Nevertheless, his model is essentially equivalent to the traditional Turing machine. In 1982, Feynman[2] discussed a quantum computer model that can simulate quantum mechanics. According to Feynman, quantum computer should have a computational mechanism that obeys the laws of quantum mechanics.

In 1985, Deutsch[3] proposed the first quantum Turing machine model with more detailed theory than Feynman's. In 1989, Deutsch[4] also presented a theory of quantum gates as another model of quantum computer: The quantum circuit model. In 1993, Yao[5] proved the equivalence of quantum circuit model and quantum Turing machine. In 1993, Bernstein and Vazirani[6] developed a universal quantum Turing machine model. They proved that quantum Turing machine has at least the same computational power as traditional Turing machine. Their research initiated the investigation of the theoretical computational aspect of quantum computer.

The first quantum algorithm was proposed by Deutsch[3] in 1985. It is for the problem of judging whether a function from $\mathbb{Z}_2$ to itself is constant. Deutsch's algorithm only needs to make one call of the function. In contrast, any classical algorithm requires two calls of the function. In 1992, Deutsch and Jozsa[7] further generalized the problem to the following: judge whether a function from $\mathbb{Z}_2^n$ to $\mathbb{Z}_2$ is constant or balanced (two-valued, one for half the domain of definition), based on the prior knowledge that the function is either constant or balanced. The new quantum algorithm still makes query to the function only once. It is deterministic and exponentially faster than any possible deterministic classical algorithm for solving this problem. The underlying problems of the two quantum algorithms may be insignificant; however, they are simple examples demonstrating the advantages of quantum superposition and entanglement.

In 1994, Simon[8] presented a quantum algorithm to solve a computational problem artificially contrived by himself. This algorithm is also exponentially faster than any possible classical algorithm for solving this problem. Simon's algorithm is more convincing than Deutsch-Jozsa's algorithm in demonstrating the supremacy of quantum computer.

Simon's algorithm inspired Shor's work on prime factorization and discrete logarithm[9], the famous Shor's algorithm. It is Shor's algorithm that shocked the world with the power of quantum computing: RSA and Diffie-Hellman cryptosystems can both be broken by a universal quantum computer.

1.2 Hidden Subgroup Problem

In 1995, Kitaev[10] noticed that the problems dealt with by Deutsch-Jozsa's algorithm and Simon's algorithm fit in a unified framework called Hidden Subgroup Problem (**HSP**). In the same paper, Kitaev proposed a method of quantum phase estimation to estimate the eigenvalues of unitary operators, which is nowadays a fundamental technique to design quantum algorithms.

Let G be a finite group, f be a map from G to a set X, such that there exists a subgroup H of G, called the hidden subgroup, with the property that $f(x) = f(y)$ whenever $xy^{-1} \in H$. The HSP is to find a generator of H, with f as the "hiding" function. In 2004, Ettinger, et al.[11]

proved that for an arbitrary group, the HSP is solvable by a polynomial number of evaluations of the hiding function. The existence of time-efficient quantum algorithms for solving the HSP for arbitrary groups is still open.

For finite Abelian groups, the HSP can be solved in polynomial time by the so-called standard method[12]. So any problem that can be transformed into an Abelian HSP can be solved efficiently in quantum computer. There are two important algorithms within this framework. The first is a polynomial-time quantum algorithm to solve Pell's equation[13], which can break Buchmann-Williams key-exchange protocol. The second is a polynomial-time quantum algorithm to compute the discrete logarithm over elliptic curve[14], which can break ECDH and ECDSA.

From 1995 to 2005, a variety of time-efficient quantum algorithms for solving the HSP of certain non-Abelian groups were proposed, such as the semi-product groups[15–19], the near-Hamiltonian groups[20], the normal subgroups[21, 22], the almost Abelian groups[23], etc. Two kinds of important non-Abelian groups are the symmetric groups and the dihedral groups. For the HSP of the two groups, no time-efficient quantum algorithm is found.

1.3 (Sub-)Exponential Speedup Quantum Algorithms for Breaking Cryptosystems

Besides Shor's algorithm and the standard method for Abelian HSP, in 2006 another quantum algorithm was proposed by van Dam, et al.[24], which broke the algebraically homomorphic cryptosystem designed in [25] (the cryptosystem can also be broken by Shor's discrete logarithm algorithm). In their algorithm, van Dam, et al. used quantum Fourier transform to solve the shifted Legendre symbol problem.

Using Simon's algorithm to break cryptosystems started around 2010-2012. Kuwakado and Morii showed that the 3-round Feistel cipher with internal permutations[26] and Even-Mansour cipher[27], can be broken with quantum superposition queries. In 2016, Santoli and Schaffner[28] used Simon's algorithm to attack symmetric-key cryptographic primitives. In the same year, Kaplan, et al.[29] discovered that the most widely used modes of operation for authentication and authenticated encryption (such as CBC-MAC, PMAC, GMAC, GCM, OCB, etc.), and CAESAR candidates of authenticated encryption schemes (such as CLOC, AEZ, COPA, OTR, POET, OMD, Minalpher, etc.), can be broken with quantum superposition queries.

Besides quantum algorithms with exponential speedup, some other algorithms with sub-exponential speedup were also proposed for quantum cryptanalysis. There are two representatives in this category.

Ajtai-Dwork cryptosystem[30, 31] is based on the computational difficulty of poly(n)-unique-SVP problem[32]. In 2004, Kuperberg[18] proposed a quantum algorithm for the HSP of dihedral groups, which solves the poly(n)-unique-SVP problem in sub-exponential time.

Finding an isogeny (a dense morphism of varieties preserving basepoints) between two elliptic curves over a finite field that have the same cardinality and endomorphism ring, is another problem with computational difficulty, for which the fastest known classical algorithm[33] takes exponential time. The isogeny-based elliptic curve cryptography[34–36] is based on the computational difficulty of this problem. By assuming the correctness of Generalized Riemann Hypothesis, in 2014 Childs, et al.[37] proposed a sub-exponential time quantum algorithm to compute the isogenies.

1.4 Grover's Search Algorithm

In 1996, an important quantum algorithm for unstructured search was invented. This is Grover's algorithm[38], which achieves quadratic speedup over classical search algorithms.

Given a finite set Ω and an oracle to query an element of Ω, let there be a nonempty subset $\mathcal{M}$ of Ω composed of marked items. Grover's algorithm can find an item of $\mathcal{M}$ by applying $O(\sqrt{|\Omega|/|\mathcal{M}|})$ times of the oracle. In 1997–1999, it was proved that Grover's algorithm is optimal for the search problem[39, 40]. When the number of marked items is known, then Grover's algorithm can be modified to be deterministic[41].

Grover's algorithm provides polynomial speedup for a large variety of important problems, such as the 3-SAT problem[42–44], the lattice shortest vector problem[45], Hamiltonian cycle problem[12], multivariate quadratic equation solving[46], weight decision[47], matrix product computing over semi-ring[48], finding the minimum of a data set[49, 50], etc. As an application of Grover's algorithm, in 1998 Brassard, et al.[51] proposed a quantum algorithm for the counting problem to estimate the number $|\mathcal{M}|$ of marked items in set Ω. It returns an integer k such that $|1-\frac{k}{|\mathcal{M}|}|\leqslant\varepsilon$ by applying $O(\varepsilon^{-1}\sqrt{|\Omega|/\mathcal{M}})$ times of the oracle. In 2000, Brassard, et al.[52] generalized Grover's algorithm to a general technique in quantum algorithms called amplitude amplification.

In the field of quantum cryptanalysis, in 1998 Brassard, et al.[53] applied Grover's algorithm to solve the collision problem, and achieved cubic speedup over classical algorithms. Their algorithm is also optimal to this problem[54]. As a consequence, attacking cryptographic protocols built on collision-resistant hash functions by quantum computer is faster. For example, breaking SHA-1 for a 160-bit hash function requires 2^{63} operations on a classical computer[55, 56]; the number of operations is reduced to approximately $2^{53.33}$ on a quantum computer.

NTRUEncrypt[57] is a public-key encryption system that is immune to attacks by Shor's algorithm. It is available as a free-for-noncommercial-use library from Security Innovation. In 2015, Fluhrer[58] applied Grover's algorithm to attack the encryption performed by this library. For (EES401EP2, EES439EP1, EES593EP1, EES743EP1), the key recovery attacks by classical computer vs. quantum computer require the following operations respectively: (2^{112} vs. 2^{104}, 2^{128} vs. 2^{112}, 2^{192} vs. 2^{137}, 2^{256} vs. 2^{197}); for plaintext recovery attacks, the data are (2^{112} vs. 2^{80}, 2^{128} vs. 2^{80}, 2^{192} vs. 2^{128}, 2^{256} vs. 2^{128}).

1.5 Quantum Walk

Random walk is a random process that models a walker on the vertices (or distribution of vertices) of a graph. At every step of the process, the walker chooses a neighbor of the current vertex at random and moves to that neighbor. It is a powerful classical algorithmic tool for searching and sampling.

As the quantum counterpart, quantum walk is not only a powerful framework for designing quantum algorithms, but a universal model of quantum computation[59]. Many important problems, such as graph collision problem[60], single-source shortest path[61–64], search on the grid[65–68], NAND tree evaluation[69–73], forbidden subgraph property[74], triangle finding[60, 75, 76], subset sum problem[77], element distinctness problem[78], group commutativity test[79], matrix product verification[80], Boolean matrix multiplication[81], algebraic property test[82], etc., can be solved by quantum walk more efficiently by constructing suitable graphs.

There are two models of quantum walk: Discrete-time and continuous-time. A discrete-time quantum walk consists of two quantum systems: A walker and a coin, together with an evolution operator which works only in discrete-time steps. A continuous-time quantum walk consists of a walker and a Hamiltonian operator of the system that works any time.

In 1985, when Feynman proposed the concept of quantum computer[83], he made another proposal that was later interpreted as a continuous-time quantum walk[84]. The definition of continuous-time quantum walk was given by Farhi, et al.[85] in 1998. As for discrete-time quantum walk, in 1996, Meyer's work[86] on quantum cellular automata already involved a walker in quantum superposition. In 2000, Nayak, et al.[87] gave the definition of discrete-time quantum walk by introducing the coin state. The relationship between continuous- and discrete-time quantum walks was investigated by Strauch[88] and Childs[89]. The results show that continuous-time quantum walk can be obtained as a limit of discrete-time quantum walks.

Early research works focused on investigating the difference of quantum walk from random walk. In [87, 90, 91], two features different from random walk were disclosed. The first is the quantum mixing time describing the speed of convergence to the uniform distribution, and the second is the quantum hitting time describing the speed of reaching the marked set. Sometimes, the mixing and hitting time of a random walk can be respectively quadratically and exponentially longer than their quantum counterparts.

For the search problem on the hypercube, a quantum algorithm based on discrete-time quantum walk was proposed by Shenvi, et al.[92] in 2003. The algorithm is quadratically faster than classical algorithms for the same problem. For the traverse problem on glued tree graphs, a quantum algorithm based on continuous-time quantum walk was proposed by Childs, et al.[93, 94] in 2002–2003. The algorithm is exponentially faster than any (not necessarily random walk based) classical algorithm.

One of the most important quantum walk algorithms was proposed by Ambainis[78] in 2007, which is an optimal quantum algorithm for solving the element distinctness problem[54]. Later in the same year, Szegedy[95] generalized Ambainis' idea and proposed a general framework of quantum walk on edges, which nowadays plays an important role in spatial search. For any symmetric and ergodic Markov chain, Szegedy's framework achieves quadratic speedup over classical algorithms for the detection problem.

Besides Szegedy's framework, another commonly used quantum walk framework on edges was proposed by Magniez, et al.[96] in 2011. This is the well-known **MNRS** framework. Based on this framework, many fast quantum algorithms were discovered, such as the algorithms for triangle finding[60], group commutativity test[79], algebraic property test[82], etc.

To obtain higher speedup, a quantum walk can be constructed inside another. This idea leads to the so-called nested quantum walk framework[61]. In 2013, Childs, et al.[97] proposed a nested quantum walk algorithm to solve the element 3-distinctness problem, which is better than applying Ambainis' element distinctness algorithm directly to element 3-distinctness problem.

1.6 Hamiltonian Simulation

Hamiltonian simulation is to find a quantum circuit that performs the unitary operation $\mathrm{e}^{-\mathrm{i}Ht}$ with error at most ε, where Hamiltonian H is an $n\times n$ Hermitian matrix with sparsity d, and t is the time variable. Hamiltonian simulation is not only for simulating quantum systems, but also an important subroutine in designing fast quantum algorithms[69, 93, 98, 99].

In the black-box oracle model, there is an oracle to query the entries of a Hamiltonian matrix. This model defines the query complexity of a Hamiltonian simulation.

The first quantum algorithm for simulating local Hamiltonians was given by Lloyd[100] in 1996, where the Hamiltonian is assumed to have a tensor product structure, so that the first order Lie product formula[101] can be used. The algorithm has query complexity $O\big((\text{poly}\log n)(\|H\|_{\text{sp}}t)^2/\varepsilon\big)$, where the *spectral norm* of a matrix A is defined as follows:

$$\|A\|_{\text{sp}} := \max_{\|\boldsymbol{x}\|=1} \|A\boldsymbol{x}\|. \tag{1.1}$$

In this paper, the norm $\|\ \|$ refers to the L_2-norm (Frobenius norm) of a vector or matrix: for any vector $\boldsymbol{v}=(v_1,\cdots,v_n)^{rmT}$ and matrix $A=(a_{ij})$,

$$\|\boldsymbol{v}\| := \sqrt{\sum_i |v_i|^2}, \qquad \|A\| := \sqrt{\sum_{i,j} |a_{ij}|^2}. \tag{1.2}$$

In 2003, Aharonov and Ta-Shma[102] generalized Lloyd's algorithm substantially to simulate sparse Hamiltonians, by replacing the requirement of tensor product structure with two requirements: 1) H is sparse, 2) there is an efficient method to calculate the nonzero entries in any column of the Hamiltonian matrix. The latter requirement is called the *sparse model assumption*, which is also satisfied by the Hamiltonian with tensor product structure in Lloyd's algorithm. Aharonov, et al.'s algorithm has query complexity $O\big(\text{poly}(d,\log n)(\|H\|_{\text{sp}}t)^{3/2}/\sqrt{\varepsilon}\big)$.

By applying the k-th order Lie product formula, in 2007 Berry, et al.[103] improved the result of Aharonov, et al. to $\widetilde{O}\big(5^{2k}d^{4+\frac{1}{2k}}t(t/\varepsilon)^{\frac{1}{2k}}\big)$. In 2010, Childs and Kothari[104] further improved the result of Berry, et al. by reducing the exponent of d to $3+\frac{1}{2k}$.

Besides the Lie product formula, quantum walk can also be used in Hamiltonian simulation. In 2012, Berry and Childs[105] proposed a quantum algorithm for Hamiltonian simulation, with query complexity $O\big(d\|H\|_{\max}t/\sqrt{\varepsilon}\big)$, where $\|H\|_{\max} := \max_{i,j}|H_{ij}|$ is the maximum norm.

By Taylor expansion, $\mathrm{e}^{-\mathrm{i}Ht}$ can be approximated by a polynomial in H. So the third approach to simulate H is to implement a matrix polynomial approximating $\mathrm{e}^{-\mathrm{i}Ht}$. Following this approach, in 2015 Berry, et al.[106] proposed an algorithm with query complexity

$$O\left(d^2\|H\|_{\max}t\left(\log\frac{t}{\varepsilon}\right)^2\Big/\left(\log\log\frac{t}{\varepsilon}\right)\right). \tag{1.3}$$

The algorithm achieves exponential speedup in parameter $1/\varepsilon$. Later in the same year, in [107] the dependence on d in (1.3) was decreased to be linear.

In conclusion, for any sparse Hamiltonian matrix (where $d = O(\log n)$), the corresponding Hamiltonian simulation is efficient. Currently the best algorithm for sparse Hamiltonian simulation has query complexity $O\big(d\|H\|_{\max}t + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\big)$, which was proposed by Low and Chuang[108] in 2016 based on quantum signal processing and block-encoding[109, 110].

For dense Hamiltonians (where $d = O(n)$), in 2010 Childs and Kothari[111] showed that there is no general Hamiltonian simulation algorithm that uses only $O\big(\mathrm{poly}(\|H\|_{\mathrm{sp}}t, 1/\varepsilon, \log n)\big)$ queries. In 2016, Rebentrost, et al.[112] presented a quantum algorithm to simulate low-rank dense Hamiltonians with query complexity $O\big(t^2n^2\|H\|_{\max}^2/\varepsilon\big)$.

Based on the memory model of storing a matrix with binary tree[113], in 2018 Wang and Wossnig[114] proposed a quantum algorithm for general dense Hamiltonian simulation by singular value estimation[113] and linear combination of unitaries[115]. The time complexity is

$$O\big(t\sqrt{n}\|H\|_{\mathrm{sp}}\,\mathrm{poly}\log(n, t\|H\|_{\mathrm{sp}}, 1/\varepsilon)\big). \tag{1.4}$$

1.7 Quantum Linear Solver

Linear system solving is a basic computational task. Given a linear system $A\boldsymbol{x} = \boldsymbol{b}$ where A is an $M \times M$ complex matrix with condition number κ, by quantum phase estimation and Hamiltonian simulation, a quantum version of the SVD of matrix A can be obtained. Based on this idea, in 2009 Harrow, Hassidim and Lloyd[99] proposed the first quantum algorithm to solve sparse linear system. This is the well-known **HHL** algorithm. It has time complexity $O(\kappa^2(\log M)/\varepsilon)$, where ε is the precision. The algorithm is exponentially faster than classical algorithms for sparse linear system solving.

In 2012, Ambainis[116] proposed an improvement to decrease the complexity by reducing the dependence on parameter κ to be linear. In 2017, Childs, et al.[115] further decreased the complexity by reducing factor $1/\varepsilon$ to $\log(1/\varepsilon)$, using the fast Hamiltonian simulation in [107].

A linear system with large condition number is ill-posed. A classical method to cope with ill-posed linear system is to use preconditioner[117]. In 2013, Clader, et al.[118] proposed a preconditioned quantum algorithm to solve ill-posed linear system.

Quantum linear solver is also studied in the block-encoding framework[109]. When the block-encoding is efficient, the complexity is linear in the condition number and logarithmic in the precision. By singular value estimation[113], in 2018 Wossnig, et al.[119] proposed a quantum algorithm to solve dense linear system, with quadratic speedup over classical algorithms.

Quantum linear solver can be extended to solve nonlinear systems. In 2017, Chen and Gao[120] proposed a quantum algorithm to solve Boolean polynomial systems, by reducing such a system to an equivalent linear system over the complex numbers with Macaulay matrix technique. In 2018, Chen, et al.[121] further extended the algorithm to solve polynomial equations over finite fields.

1.8 Quantum Machine Learning

The study of quantum machine learning boomed after the discovery of HHL algorithm[122, 123]. The first application of HHL algorithm in machine learning is Wiebe, et al.'s work[124] on data fitting, which was followed by a substantial amount of work on the same topic[109, 113, 125, 126].

In 2013, Lloyd, et al.[127] proposed a quantum algorithm for supervised and unsupervised classification, with exponential speedup. In 2014, Lloyd, et al.[128] proposed a method for quantum principal component analysis. In 2014, Rebentrost, et al.[129] applied HHL algorithm to achieve exponential speedup in least-square support vector machine.

In 2017, Rebentrost, et al.[130] proposed a quantum Hopfield neural network model. In this model, the weight matrix is represented as a density matrix that can be simulated by the method in [128]. The learning of the weights by minimizing the energy function is then reduced to linear system solving.

By now quantum algorithms are found many important applications in machine learning: neural network[131–136], pattern recognition[137–139], Bayesian theory[140, 141], hidden Markov model[142, 143], deep learning[144, 145], Boltzmann machine[146–148], etc.

1.9 Robust Quantum Computing and Quantum Complexity Theory

Adiabatic quantum computing (**AQC**) is a class of procedures for solving optimization problems with quantum computer. To solve a problem with AQC, suppose that an appropriate Hamiltonian H_1 can be found, so that the solution to the problem is represented as a ground state of the Hamiltonian. An adiabatic algorithm begins with a system in the ground state of a known and easily implementable Hamiltonian H_0. A path H_t for $t \in [0, 1]$ is chosen between the initial Hamiltonian H_0 and the final Hamiltonian H_1, and the Hamiltonian is gradually perturbed to follow this path. The theory of adiabatic quantum computation is established on the Adiabatic Theorem[149], which states that as long as the path is traversed slowly enough, the system will remain in the ground state, and in the end it will be in the solution state.

AQC was first introduced by Farhi, et al.[150] in 2000. In 2001, Childs, et al.[151] showed that AQC has some inherent protection against decoherence, meaning that it may be a particularly good model for designing robust quantum algorithms. In 2002, Roland and Cerf[152] showed how to recapture Grover's algorithm and its optimality proof within the adiabatic context. In 2007, Aharonov, et al.[153] developed a model for AQC and proved its computational equivalence with quantum circuit model.

Initial interests in AQC concentrated on the possibility of using adiabatic methods to design quantum algorithms for some NPC problems[154–158]. Recently, AQC is used to solve some problems in machine learning[159, 160], graph theory[161, 162], etc.

In 1997, Kitaev[163] proposed topological quantum computing, a more speculative approach to quantum computing with excellent robustness. In 2003, Freedman, et al.[164] investigated the relationship between topological quantum field theory and topological quantum computing, with the result that on one hand, quantum computer can efficiently simulate topological quantum field theory, on the other hand, topological quantum field theory essentially captures the power of quantum computing.

One important application of topological quantum computing is a polynomial-time quantum algorithm for the approximation of Jones polynomial[165], which itself is a BQP-complete problem[164]. This work inspired the creation of many efficient quantum algorithms, for problems such as approximation of the Tutte polynomial of a planar graph[166], evaluation of the Jones polynomial on a generalized closure of a braid[167], computation of additive approximation of a tensor network[168], etc.

In computational complexity theory, BPP (bounded-error probabilistic polynomial time) refers to the class of computational problems that can be solved efficiently by a classical computer with success probability larger than 2/3. **BQP** (bounded-error quantum polynomial time) is the quantum version of BPP. In 1993, Bernstein and Vazirani[6] proved the famous result that BQP $\subseteq$ PSPACE, the latter being all decision problems that can be solved using polynomial space. It is conjectured[169] that BQP $\cap$ NPC $= \emptyset$, and NP $\not\subseteq$ BQP. It is also suspected[170] that P $\subset$ BQP.

In September 2018, Mahadev[171] proposed the first measurement protocol to show that a BPP machine can interact with a BQP machine in order to verify BQP computations, based on the assumption that the verifier may use post-quantum cryptography that the BQP prover cannot break.

1.10 Hardware Development and Quantum Algorithm Demonstration

In 1995, Cirac and Zoller[172] developed a trapped ion quantum computer and realized the CNOT gate. In 1997, Gershenfeld and Chuang[173] developed a 2-qubit NMR quantum computer. In 1998, Loss and Divincenzo[174] made a new implementation of quantum computer based on quantum dot.

In 2001, IBM developed a 7-qubit NMR quantum computer, and implemented Shor's algorithm[175] on it. In 2004, Pan's group[176] demonstrated the first 5-photon entanglement for universal quantum error correction. In 2006, the Institute for Quantum Computing and the Perimeter Institute for Theoretical Physics in Waterloo, as well as MIT, benchmarked a 12-qubit quantum computer[177]. In 2007, D-Wave demonstrated a 28-qubit quantum annealing computer[178]. In 2008, D-Wave produced a 128-qubit computer chip[179].

In 2011, Monz, et al.[180] reported the creation of GHZ states with up to 14 qubits. In 2015, D-Wave announced the breaking of the 1000-qubit barrier[181]. In 2016, Google simulated a hydrogen molecule with an array of 9 superconducting qubits developed by the Martinis group and UCSB[182].

In 2017, IBM unveiled a 17-qubit quantum computer[183], and Intel confirmed the development of a 17-qubit superconducting test chip[184]. IBM also announced an industry-first initiative to build a universal quantum computing system, called "IBM Quantum Experience"[185], that enables developers and programmers to begin building interfaces between its existing 5-qubit cloud-based quantum computer and classical computers. Coles, et al.[186] implemented 20 quantum algorithms on IBM Quantum Experience. In the same year, Pan, et al.[187] made the first 10 entangled photons. Later in this year, IBM revealed a working 50-qubit quantum computer that can maintain its quantum state for 90 microseconds[188], and D-Wave released a 2000-qubit quantum annealing computer[189].

In 2018, Intel confirmed the development of a 49-qubit superconducting test chip[190]. In the same year, Google announced the creation of a 72-qubit quantum chip[191].

With the boost of quantum computer hardware, quantum algorithms are tested by experiments of increasing (but still very small) scale. Table 1 collects some experiments on three most famous quantum algorithms: Shor's algorithm, Grover's algorithm, and HHL algorithm.

Table 1 Experimental demonstration of quantum algorithms

Algorithm	Problem solved	# qubits: Technology
Shor	factorizing 15	4: photonic[192] (2007)
		6: photonic[193] (2007)
		7: NMR[175] (2001)
	factorizing 21	5: photonic[194] (2012)
Grover	searching 1 marked item in 4 items	2: NMR[195] (1998)
	searching 1 marked item in 8 items	3: NMR[196] (2000)
HHL	solving 2×2 linear system	3: photonic[197] (2014)
		4: photonic[198] (2013)
		4: NMR[199] (2014)

In the development of quantum algorithms, some techniques are so important that they become universal subroutines of many efficient algorithms. These techniques include quantum phase estimation, quantum amplitude amplification, quantum singular value estimation, linear combination of unitaries (**LCU**), classical vector input, quantum linear solver, swap test, Grover search, Szegedy's quantum walk, MNRS quantum walk, etc. More detailed description and analysis of these subroutines and their applications are the content of the rest of this paper.

In Section 2, some basic terminology in quantum algorithms are presented. In Section 3, first a detailed description of quantum phase estimation is given, then order finding and swap test are introduced as applications of quantum phase estimation, and then LCU and its application in quantum state preparation are investigated. In Section 4, first two major quantum linear solvers are reviewed: HHL algorithm and singular value estimation, then an algorithm to solve Boolean equation system is introduced as an application of HHL algorithm, and then three aspects of quantum machine learning are investigated as applications of quantum linear solvers: Supervised classification, linear regression, and support vector machine. In Section 5, basics of quantum walk are introduced, together with applications in the search problems on glued tree graphs and Boolean hypercubes. In Section 6, several frameworks of quantum search are reviewed, together with their applications in collision problem, element distinctness problem, and spatial search. In Section 7, several open problems and emerging research directions in quantum algorithm design are listed.

Some earlier reviews on the development of quantum algorithms include [200–204]. Comprehensive introductions to quantum computing and quantum algorithm can be found in [12, 205–207]. Previous surveys of quantum walk include [42, 208–212]. A comprehensive review of

Hamiltonian simulation is referred to [213]. Some reviews of quantum machine learning include [68, 214–219]. A recent review of adiabatic quantum computation is [220]. A website that collects many quantum algorithms can be found here: [221].

2 Basic Terminology in Quantum Algorithms

The following are standard notations in computational complexity. Let f, g be positive functions in $n \in \mathbb{N}$. Then

- $f = O(g)$ if there exist constants $c > 0$ and $n_0 > 0$ such that $f(n) \leq cg(n)$ for all $n \geq n_0$.
- $f = \Omega(g)$ if there exist constants $c > 0$ and $n_0 > 0$ such that $f(n) \geq cg(n)$ for all $n \geq n_0$.
- $f = \Theta(g)$ if both $f = O(g)$ and $f = \Omega(g)$.
- $\widetilde{O}$, $\widetilde{\Omega}$, $\widetilde{\Theta}$ differ from O, Ω, Θ respectively only by a logarithmic factor.
- $\mathrm{poly}(n)$ (or $\mathrm{poly}\, n$) refers to a polynomial in n.
- $\log(n_1, \cdots, n_k)$ refers to a product of logarithmic factors in $n_1, \cdots, n_k$ respectively.

In quantum computing, the *bra-ket notation* or *Dirac notation* is standard for describing quantum states. In this notation, a complex vector $\boldsymbol{a} = (a_1, \cdots, a_n)^{\mathrm{T}}$ when representing a quantum state, is written as $|\boldsymbol{a}\rangle$, and its conjugate transpose $\boldsymbol{a}^{\dagger}$ is written as $\langle \boldsymbol{a}| = (\overline{a}_1, \cdots, \overline{a}_n)$.

For two vectors $\boldsymbol{a} = (a_1, \cdots, a_m)^{\mathrm{T}}$ and $\boldsymbol{b} = (b_1, \cdots, b_n)^{\mathrm{T}}$, the $m \times n$ matrix

$$(a_i b_j)_{i=1,2,\cdots,m, j=1,2,\cdots,n}$$

is denoted by $|\boldsymbol{a}\rangle\langle \boldsymbol{b}|$. When $m = n$, then $\langle \boldsymbol{a}|\boldsymbol{b}\rangle$ denotes $\boldsymbol{a} \cdot \boldsymbol{b}$. The tensor product $\boldsymbol{a} \otimes \boldsymbol{b}$ is a column vector of dimension mn, and is written as

$$|\boldsymbol{ab}\rangle = |\boldsymbol{a}, \boldsymbol{b}\rangle := |\boldsymbol{a}\rangle \otimes |\boldsymbol{b}\rangle \tag{2.1}$$

when representing a quantum state.

While the fundamental information unit of classical computer is "bit", the fundamental unit of quantum computer is quantum bit, or *qubit* for short. A qubit is a unit complex linear combination of two orthonormal basis states (ground states) $|0\rangle, |1\rangle$ that correspond to classical bits $0, 1$ respectively. So a qubit has the form

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha|0\rangle + |0\rangle^{\perp}, \tag{2.2}$$

where $|0\rangle^{\perp}$ is a shorthand notation of a linear combination of quantum states orthogonal to $|0\rangle$.

A quantum state with n qubits is a normalized complex linear combination of several n-fold tensor products of $|0\rangle, |1\rangle$. It is of the form (binary form)

$$|\psi\rangle = \sum_{i_1, \cdots, i_n \in \{0,1\}} \alpha_{i_1, \cdots, i_n} |i_1 \cdots i_n\rangle, \tag{2.3}$$

where $\alpha_{i_1,\cdots,i_n} \in \mathbb{C}$ is the *amplitude* of $|\psi\rangle$ in basis state $|i_1\cdots i_n\rangle$. The basis states $|i_1\cdots i_n\rangle$ are orthonormal.

By binary expression of integers, i.e., $x = \sum_{j=0}^{n-1} i_j 2^j$ for any integer $0 \le x < 2^n$, where the $i_j \in \{0,1\}$, (2.3) can be written as the following integer form:

$$|\psi\rangle = \sum_{x=0}^{2^n-1} \alpha_x |x\rangle. \tag{2.4}$$

When there is no need to make explicit the number of qubits, quantum state $|0\rangle^{\otimes n}$ in binary form is usually written as $|0\rangle$ in integer form. For example, for $N = 2^n$, let $|\phi\rangle = (x_1,\cdots,x_N)^{\mathrm{T}}$ be the coordinate form of $|\phi\rangle$ with respect to the basis states $|0\rangle,\cdots,|N-1\rangle$ (in integer form) of $\mathbb{C}^N$. Let $|0\rangle,|1\rangle$ (in binary form) be the basis states of $\mathbb{C}^2$. Then in the space $\mathbb{C}^2\otimes\mathbb{C}^N$,

$$|0\rangle\otimes|\phi\rangle = (x_1,\cdots,x_N,0,\cdots,0)^{\mathrm{T}}, \qquad |1\rangle\otimes|\phi\rangle = (0,\cdots,0,x_1,\cdots,x_N)^{\mathrm{T}}. \tag{2.5}$$

Like classical computer, quantum computer uses *quantum register* composed of multiple qubits. For quantum states $|\psi_1\rangle,\cdots,|\psi_k\rangle$, in the tensor product state

$$|\psi_1\psi_2\cdots\psi_k\rangle = |\psi_1\rangle|\psi_2\rangle\cdots|\psi_k\rangle, \tag{2.6}$$

the i-th register $(1 \le i \le k)$ refers to position i of the tensor product, and $|\psi_i\rangle$ is called the *state value* of the i-th register in (2.6). Sometimes the i-th register is simply called register $|\psi_i\rangle$.

A quantum state is always represented by a unit vector of an appropriate dimension. For a vector $\boldsymbol{a} = (a_0,\cdots,a_{n-1})^{\mathrm{T}} = \sum_{i=0}^{n-1} a_i\boldsymbol{e}_i$ where the $\boldsymbol{e}_i$ are an orthonormal basis,

$$|\boldsymbol{a}\rangle := \frac{1}{\|\boldsymbol{a}\|}\sum_{i=0}^{n-1} a_i|\boldsymbol{e}_i\rangle = \frac{1}{\|\boldsymbol{a}\|}\sum_{i=0}^{n-1} a_i|i\rangle, \tag{2.7}$$

where $|i\rangle := |\boldsymbol{e}_i\rangle$ represents the basis state by its subscript.

A *measurement* to a quantum state can be understood as a projection from the vector representation of the quantum state to a set of fixed vectors (called measurement basis) whose span contains the quantum state to be measured. In a measurement at some register i, the projection is only for the state value at register i, while all other registers remain unaffected.

After measurement, a quantum state collapses to a state of the measurement basis. For a measurement at some register i, only the state value at register i collapses to a state of the measurement basis, while all other registers remain unchanged. For example, when measuring (2.3) with the $|i_1\cdots i_n\rangle$ as the default measurement basis, then any of the basis states $|i_1\cdots i_n\rangle$ can be the result of measurement; the probability to obtain a specific result $|j_1\cdots j_n\rangle$ is the squared norm of the corresponding amplitude: $|\alpha_{j_1,\cdots,j_n}|^2$. When the measurement occurs at register k of (2.3) for some $1 \le k \le n$, suppose $|i_k\rangle$ is collapsed to $|0\rangle$, then the result of measurement of (2.3) is (after deleting register k):

$$\sum_{i_1,\cdots,\check{i}_k,\cdots,i_n\in\{0,1\}} \alpha_{i_1,\cdots,i_{k-1},0,i_{k+1},\cdots,i_n}|i_1\cdots\check{i}_k\cdots i_n\rangle \tag{2.8}$$

up to normalization.

Since quantum states are normalized, quantum computer only allows unitary operations. Some frequently used unitary operators include:

(i) Complex rotation $R_{a,b}$ in a 2-space with fixed orthonormal basis $\boldsymbol{e}_1, \boldsymbol{e}_2$, where $a, b \in \mathbb{C}$ such that $|a|^2 + |b|^2 = 1$.

The matrix form of the rotation with respect to the basis is

$$\begin{bmatrix} a & -\overline{b} \\ b & \overline{a} \end{bmatrix}. \tag{2.9}$$

It changes $|\boldsymbol{e}_1\rangle$ into $a|\boldsymbol{e}_1\rangle + b|\boldsymbol{e}_2\rangle$. When a is given, a simple choice of b is $\sqrt{1-|a|^2}$.

(ii) Reflection with respect to a subspace of $\mathbb{C}^N$.

A reflection with respect to an m-space W of $\mathbb{C}^N$ is the unitary transformation preserving all vectors of W while reversing all vectors of $W^\perp$. If W is spanned by orthonormal basis vectors $\{|\boldsymbol{w}_i\rangle,\ i = 1, 2, \cdots, m\}$, then

$$R_W = 2\sum_{i=1}^{m} |\boldsymbol{w}_i\rangle\langle\boldsymbol{w}_i| - I, \tag{2.10}$$

where I is the identity matrix. The reflection with respect to $W^\perp$ is $R_{W^\perp} = -R_W$.

When $N = 2$, reflection $R_{|0\rangle}$ is called Pauli-Z transformation in the 2-space spanned by $|0\rangle, |1\rangle$.

(iii) Quantum Fourier transform: the matrix form is

$$\mathcal{F}_N = \frac{1}{\sqrt{N}}(\mathrm{e}^{2\pi\mathrm{i}xy/N})_{x,y=0,1,\cdots,N-1}. \tag{2.11}$$

(iv) Hadamard transform (Hadamard gate) $\mathcal{H}^{\otimes n}$, where $\mathcal{H} = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$.

If $|x\rangle$ is a basis state, then $\mathcal{H}^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{x\cdot y}|y\rangle$, where $x\cdot y$ is the inner product of the binary expressions of integers x, y taken as vectors of dimension n.

When $n = 1$, denote

$$|+\rangle := \mathcal{H}|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad |-\rangle := \mathcal{H}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{2.12}$$

When $x = 0$, then

$$\mathcal{H}^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}|y\rangle =: |\varpi_{2^n}\rangle. \tag{2.13}$$

It is the uniform superposition of all n-qubit ground states.

$\mathcal{H}^{\otimes n}$ has circuit implementation complexity $O(n)$. It is often written as $\mathcal{H}$ when the multiplicity n of the tensor product is omittable.

(v) Grover's diffusion: $R_{|\varpi_{2^n}\rangle}$, with $|\varpi_{2^n}\rangle$ given by (2.13).

It has circuit implementation complexity $O(n)$, because of the following decomposition:

$$R_{|\varpi_{2^n}\rangle} = \mathcal{H}^{\otimes n}(2|0\rangle^{\otimes n}\langle 0|^{\otimes n} - I)\mathcal{H}^{\otimes n}. \tag{2.14}$$

(vi) Elimination of register.

The inverse of Hadamard transform is itself. So for any $|\phi\rangle, |\psi\rangle$,

$$\mathcal{H}\left(\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1} |\phi\rangle\, |y\rangle|\psi\rangle\right) = |\phi\rangle|0\rangle|\psi\rangle. \tag{2.15}$$

(2.15) is called the *elimination of state value* $|y\rangle$ for all $y \neq 0$ from the second register.

In many cases, the constant state value $|0\rangle$ is omitted for succinctness, and the result of (2.15) becomes $|\phi\rangle|\psi\rangle$. In this notation, (2.15) is called the elimination of the second register, or *elimination of register* $|y\rangle$ for all $0 \le y < 2^n$.

(vii) Quantum implementation of classical map.

Let $f : \mathbb{Z}_{2^n} \longrightarrow \mathbb{Z}_{2^m}$ be a map. The *quantum implementation* of f refers to a unitary transformation generating the following state from initial state $|0\rangle^{\otimes n}|0\rangle^{\otimes m}$:

$$\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle. \tag{2.16}$$

The quantum implementation can be realized by $\mathcal{U}_f\mathcal{H}$, in which $\mathcal{H}$ is the Hadamard transform, and $\mathcal{U}_f$ is the following unitary operator:

$$\mathcal{U}_f : |x, y\rangle \mapsto |x, y \oplus f(x)\rangle, \quad \forall x \in \mathbb{Z}_{2^n},\ y \in \mathbb{Z}_{2^m}, \tag{2.17}$$

where $\oplus$ denotes the bit-wise XOR.

(viii) Control transformation

$$C = \sum_{j=0}^{k-1} |j\rangle\langle j| \otimes C_j, \tag{2.18}$$

where each C_j is a unitary operator.

(2.18) is called the action of C_j (on the second register) *controlled by register* $|j\rangle$. For any $\alpha_{j,x} \in \mathbb{C}$,

$$C\left(\sum_{j,x} \alpha_{j,x}|j\rangle|x\rangle\right) = \sum_{j,x} \alpha_{j,x}|j\rangle\, C_j|x\rangle. \tag{2.19}$$

With a black-box oracle to query the entries of a unitary matrix, any sparse unitary operator U can be efficiently implemented in quantum computer[105]. The reason is that for any quantum state $|\phi\rangle$,

$$\mathrm{e}^{-\mathrm{i}H\pi/2}|1\rangle \otimes |\phi\rangle = -\mathrm{i}|0\rangle \otimes U|\phi\rangle, \tag{2.20}$$

where $H = \left[\begin{smallmatrix} 0 & U \\ U^\dagger & 0 \end{smallmatrix}\right]$ is a sparse Hermitian matrix. The notation "†" refers to the *matrix conjugate transpose*. Since $\mathrm{e}^{-\mathrm{i}H\pi/2}$ can be implemented efficiently by sparse Hamiltonian simulation, so can unitary operator U.

A quantum algorithm contains an input, a series of unitary transformations, and an output obtained by measurement. If Step 1 to Step k are the composition of unitary transformations $U_1, \cdots, U_k$, then "undo" step 1 to step k refers to the unitary transformation $(U_1 \cdots U_k)^{-1}$.

Any quantum algorithm is probabilistic. If the probability of getting the desired result is P, then by quantum amplitude amplification[52], it suffices to run the algorithm $O(1/\sqrt{P})$ times, each time ending with measurement. This is already quadratic speedup over classical probabilistic algorithms.

The *universal gates* (or elementary gates) is a set of unitary operators acting on one or two qubits. It is required that any unitary operator can be approximated to some precision by a finite composition of elementary gates. The number of elementary gates in the composition measures the efficiency of the unitary operator.

The *circuit complexity* (or time complexity, or gate complexity) of a quantum algorithm refers to the lower bound of the number of elementary gates in a quantum circuit to realize the unitary operators in the quantum algorithm. A computation is said to be *efficient*, if the number of gates is polynomial in the number of qubits used in the computation. A result of Solovay and Kitaev[12, 222, 223] states that any unitary operator that can be realized efficiently by a set of universal gates can also be realized efficiently by any other universal gates.

Besides circuit complexity, another evaluation of quantum algorithm is query complexity. If in a quantum algorithm involving a given function f, the computation load concentrates on the number of visits to f, then the number of visits to f in the algorithm is called the *query complexity*. In a quantum walk algorithm, the query complexity is also referred to as the *step complexity* (one query per step).

With the above terminology, some key concepts in designing quantum algorithms can be re-introduced technically below:

(I) **Quantum phase estimation** Let U be a unitary operator. Then its eigenvalues are of the form $\mathrm{e}^{\mathrm{i}\theta_j}$, where $0 \leq \theta_j < 2\pi$. The θ_j are called the *eigenphases* of U.

Given unitary operator U and eigenvector $|\boldsymbol{u}\rangle$, estimating the corresponding eigenphase $\theta \in [0, 2\pi)$ is called quantum phase estimation.

(II) **Quantum amplitude amplification** Let $|\psi\rangle = \alpha|x\rangle + |x\rangle^\perp$ be a quantum state, where $\alpha \in \mathbb{C}$ is the amplitude of $|\psi\rangle$ in basis state $|x\rangle$. By measuring $|\psi\rangle$, one gets state $|x\rangle$ with probability $|\alpha|^2$, which can be a very small number.

With Grover search, by $O(1/|\alpha|)$ times of preparing $|\psi\rangle$ and then making measurement, state $|x\rangle$ can be obtained with high probability. This subroutine of improving the probability of a component basis state in a quantum state by repetition following Grover's algorithm, is called *quantum amplitude amplification*.

(III) **Block encoding** Realizing linear transformation is a fundamental task in quantum computing. A block-encoding of a square matrix A is a unitary matrix U such that the top-left block of U is equal to A/t for some normalizing constant $t \geq \|A\|_{\rm sp}$.

(IV) **Linear combinations of unitaries (LCU)** Any square matrix can be written as a linear combination of unitary matrices. LCU algorithm is on quantum realization of a linear transformation given in the form of LCU.

Let $L = \sum_i \alpha_i U_i$, where the U_i are unitary, and the $\alpha_i \in \mathbb{C}$. A *quantum realization* of L is a unitary transformation mapping an arbitrary state $|\phi\rangle|0\rangle$ to

$$|\psi\rangle = \frac{1}{t} L|\phi\rangle|0\rangle + (\cdots)|0\rangle^{\perp}, \tag{2.21}$$

where t is a know parameter satisfying $t \geq \|L|\phi\rangle\|$.

(V) **Quantum preparation of classical vector** Given a complex vector $\boldsymbol{x} = (x_1, \cdots, x_n)^{\rm T}$, a *quantum preparation* of the unnormalized quantum state $\|\boldsymbol{x}\|\,|\boldsymbol{x}\rangle$ of $\boldsymbol{x}$ is a unitary transformation from initial state $|0\rangle|0\rangle$ to

$$|\psi\rangle = \frac{\|\boldsymbol{x}\|}{t}|\boldsymbol{x}\rangle|0\rangle + (\cdots)|0\rangle^{\perp}, \tag{2.22}$$

where $t \geq \|\boldsymbol{x}\|$ is a know parameter.

(VI) **Quantum eigenphase decomposition of unitary operator** Let U be an $n\times n$ unitary matrix with eigenvalues $e^{i\theta_j}$ and corresponding eigenvectors $\boldsymbol{u}_j$ for $0 \leq j \leq n-1$.

A *quantum eigenphase decomposition* of U with precision ε refers to a unitary transformation such that for any $\alpha_j \in \mathbb{C}$ satisfying $\sum_{j=1}^{n} |\alpha_j|^2 = 1$,

$$\sum_{j=0}^{n-1} \alpha_j |\boldsymbol{u}_j\rangle|0\rangle \mapsto \sum_{j=0}^{n-1} \alpha_j |\boldsymbol{u}_j\rangle|\widetilde{\theta}_j\rangle, \tag{2.23}$$

where $\widetilde{\theta}_j$ is an ε-approximate of θ_j for all $0 \leq j < n$, and both $\theta_j, \widetilde{\theta}_j \in [0, 2\pi)$.

(VII) **Quantum singular value estimation (SVE)**

Let A be an $m \times n$ matrix. Recall that if the nonzero singular values of A are $\{\sigma_i > 0,\ i = 1, 2, \cdots, r\}$, where r is the rank of A, then an SVD of A is $A = UDV^{\dagger}$, where $U = (\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m)$ and $V = (\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n)$ are unitary matrices, and $D = (d_{ij})$ is an $m \times n$ matrix whose only nonzero entries are $d_{ii} = \sigma_i$ for all $1 \leq i \leq r$. Obviously

$$\|A\|_{\rm sp} = \max_{i=1,2,\cdots,r} |\sigma_i|. \tag{2.24}$$

In the special case when A is Hermitian, then $m = n$, the nonzero eigenvalues of A are $\{\theta_i = \delta_i\sigma_i,\ i = 1, 2, \cdots, r\}$, where $\delta_i \in \{1, -1\}$ is the sign of θ_i. For any $1 \leq i \leq r$, $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are related by $\boldsymbol{v}_i = \delta_i \boldsymbol{u}_i$, both of which are eigenvectors corresponding to eigenvalue θ_i. The zero eigenvalues agree with the zero singular values, and $\boldsymbol{u}_i = \boldsymbol{v}_i$ for $r < i \leq n$.

When $m = n$, a quantum **SVE** of A is a unitary transformation with one of the following properties: For any $\alpha_i \in \mathbb{C}$ satisfying $\sum_{i=1}^{n} |\alpha_i|^2 = 1$,

$$\begin{array}{llll} 1) & \sum_{i=1}^{n} \alpha_i |\boldsymbol{v}_i\rangle|0\rangle & \mapsto & \sum_{i=1}^{n} \alpha_i |\boldsymbol{u}_i\rangle|\widetilde{\sigma}_i\rangle, \\ 2) & \sum_{i=1}^{n} \alpha_i |\boldsymbol{u}_i\rangle|0\rangle & \mapsto & \sum_{i=1}^{n} \alpha_i |\boldsymbol{v}_i\rangle|\widetilde{\sigma}_i\rangle, \end{array} \tag{2.25}$$

where $\widetilde{\sigma}_i$ is an approximate of σ_i with error bound $|\widetilde{\sigma}_i - \sigma_i| \le \varepsilon\|A\|_{\text{sp}}$.

(VIII) **Quantum SVE for non-square matrix** When $m \ne n$, the *extended Hermitian matrix* of $m \times n$ matrix A is

$$A^{\text{ext}} := \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}. \tag{2.26}$$

The nonzero eigenvalues of A^{ext} are $\{\pm\sigma_i,\ 1 \le i \le r\}$, the corresponding eigenvectors are $(\boldsymbol{u}_i^{\text{T}}, \pm\boldsymbol{v}_i^{\text{T}})^{\text{T}}$. The orthogonal complement of the r-space spanned by $\{(\boldsymbol{u}_i^{\text{T}}, \boldsymbol{v}_i^{\text{T}})^{\text{T}},\ 1 \le i \le r\}$ in $\mathbb{C}^{m+n}$ is the eigenspace of A^{ext} corresponding to eigenvalue 0, and is spanned by the $(\boldsymbol{u}_j^{\text{T}}, 0)^{\text{T}}$ for all $r < j \le m$ and the $(0, \boldsymbol{v}_k^{\text{T}})^{\text{T}}$ for all $r < j \le n$.

A quantum SVE[110, 113, 119] of A can be deduced from that of A^{ext}. It is a unitary transformation such that for any $\alpha_i, \alpha_{-i} \in \mathbb{C}$ satisfying $\sum_{i=1}^{r}(|\alpha_i|^2 + |\alpha_{-i}|^2) = 1$,

$$\sum_{i=1}^{r} \big(\alpha_i|\boldsymbol{u}_i, \boldsymbol{v}_i\rangle + \alpha_{-i}|\boldsymbol{u}_i, -\boldsymbol{v}_i\rangle\big)|0\rangle \mapsto \sum_{i=1}^{r} \big(\alpha_i|\boldsymbol{u}_i, \boldsymbol{v}_i\rangle|\widetilde{\sigma}_i\rangle + \alpha_{-i}|\boldsymbol{u}_i, -\boldsymbol{v}_i\rangle|\widetilde{-\sigma_i}\rangle\big), \tag{2.27}$$

where $\widetilde{\sigma}_i, \widetilde{-\sigma_i}$ are approximates of $\sigma_i, -\sigma_i$ respectively, with error bound $\varepsilon\|A\|_{\text{sp}}$.

(IX) **Quantum linear solver** For linear system $A\boldsymbol{x} = \boldsymbol{b}$, a quantum linear solver outputs a quantum state $|\boldsymbol{x}\rangle$, such that

$$\big\|t|\boldsymbol{x}\rangle - A^{+}|\boldsymbol{b}\rangle\big\| \le \varepsilon\|A^{+}\|_{\text{sp}} \tag{2.28}$$

for some unknown parameter $t > 0$, where A^{+} is the Moore-Penrose inverse of A.

Rescaling A does not influence (2.28). Usually A is rescaled to satisfy $\|A\|_{\text{sp}} = 1$.

(X) **Swap test** For any two quantum states $|\boldsymbol{x}\rangle, |\boldsymbol{y}\rangle$, swap test is a technique to estimate the inner product $\langle\boldsymbol{x}|\boldsymbol{y}\rangle = \boldsymbol{x}\cdot\boldsymbol{y}$ of two unit vectors $\boldsymbol{x}, \boldsymbol{y}$.

Many quantum algorithms only return the quantum state of a classical vector. Swap test is used to extract classical information from a quantum state.

3 Some Building Blocks in Quantum Algorithm Design

This section introduces two fundamental techniques in quantum algorithm design, together with several key subroutines based on them. The two techniques are phase estimation and

LCU. The former is used to extract the coordinates from a unit vector given in the form of a quantum state, and the latter is used to implement (non-unitary) linear maps and classical vectors on quantum computer. They are counterparts of the interface between classical data and quantum data.

3.1 Quantum Phase Estimation

Quantum phase estimation is among the most important techniques in quantum algorithm design. It was first proposed by Kitaev[10] in 1995 as a generalization of a key technique in Shor's algorithm[9]. Currently, it is an important subroutine of many quantum algorithms, such as quantum linear solver[99, 119], quantum principal component analysis[128], quantum singular value decomposition[112, 113], eigenvalue estimation of Hermitian matrix[224], quantum counting[51], swap test[225], Hamiltonian simulation based on quantum walk[89, 105], etc.

Let U be a unitary transformation, and $\boldsymbol{u}$ be a unit eigenvector corresponding to eigenvalue $\mathrm{e}^{2\pi\mathrm{i}\theta}$, where $0 \le \theta < 1$. Given U, the following algorithm returns a value $\widetilde{\theta} \in [0,1)$ such that $|\theta-\widetilde{\theta}| \le 2^{-(n+1)}$ for initial state $|\boldsymbol{u}\rangle$, where integer n is introduced to represent the bit precision. It is also the number of qubits needed by the algorithm to run on.

Algorithm 1 Quantum phase estimation[12]

```
1: Prepare the initial state $|\psi_1\rangle = |0\rangle^{\otimes n}|\boldsymbol{u}\rangle$ (assume the preparation is efficient).
2: Apply Hadamard transform to the first register $|0\rangle^{\otimes n}$ of $|\psi_1\rangle$. Result:
       $$|\psi_2\rangle = |\varpi_{2^n}\rangle|\boldsymbol{u}\rangle = \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}|y\rangle|\boldsymbol{u}\rangle. \quad (3.1)$$
3: Apply control transformation $\sum_{y=0}^{2^n-1}|y\rangle\langle y|\otimes U^y$ to $|\psi_2\rangle$. The effect is that the $y$-th power
   $U^y$ of $U$ is applied to $|\boldsymbol{u}\rangle$ if the first register takes state value $|y\rangle$. The result is
       $$|\psi_3\rangle = \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}|y\rangle U^y|\boldsymbol{u}\rangle = \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}\mathrm{e}^{2\pi\mathrm{i}\theta y}|y\rangle|\boldsymbol{u}\rangle. \quad (3.2)$$
4: Apply inverse quantum Fourier transform to the first register $|y\rangle$ of $|\psi_3\rangle$. Result:
       $$|\psi_4\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1}\mathrm{e}^{2\pi\mathrm{i}\theta y-2\pi\mathrm{i}\frac{xy}{2^n}}|x\rangle|\boldsymbol{u}\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}\left[\sum_{y=0}^{2^n-1}\mathrm{e}^{2\pi\mathrm{i}y(\theta-\frac{x}{2^n})}\right]|x\rangle|\boldsymbol{u}\rangle. \quad (3.3)$$
5: Perform measurement to $|\psi_4\rangle$.
```

The measurement returns a basis state $|x\rangle|\boldsymbol{u}\rangle$ for some $0 \le x < 2^n$, from which the value of x in binary expression is obtained. With high probability, the rational number $x/2^n$ is an approximate of θ to precision $2^{-(n+1)}$. This claim is established as follows[12].

Denote the error function

$$\delta(x) := \theta - \frac{x}{2^n}, \quad \forall x \in \mathbb{R}. \tag{3.4}$$

Since $0 \leq \theta < 1$, θ can be written in binary expression form as

$$\theta = \frac{\theta_1}{2} + \cdots + \frac{\theta_n}{2^n} + r_n = \frac{\theta_1 2^{n-1} + \cdots + \theta_n}{2^n} + r_n, \tag{3.5}$$

where $\theta_i \in \{0,1\}$ and $0 \leq r_n < 2^{-n}$. Denote integer

$$n_\theta := \theta_1 2^{n-1} + \cdots + \theta_n. \tag{3.6}$$

It is the integer that one hopes the algorithm to output.

Case 1 If $r_n = 0$, then $\delta(n_\theta) = 0$, and the amplitude of $|\psi_4\rangle$ in $|n_\theta\rangle|\boldsymbol{u}\rangle$ is

$$\frac{1}{2^n}\sum_{y=0}^{2^n-1} \mathrm{e}^{2\pi \mathrm{i} y(\theta - \frac{n_\theta}{2^n})} = 1. \tag{3.7}$$

So $|\psi_4\rangle = |n_\theta\rangle|u\rangle$. The result n_θ is obtained by measurement with probability 1, and the algorithm is deterministic.

Case 2 If $0 < r_n < 2^{-(n+1)}$, then $\delta(n_\theta) = r_n \in (0, 2^{-(n+1)})$, and $-\delta(n_\theta + 1) = 2^{-n} - r_n \in (0, 2^{-n})$. Between $n_\theta/2^n$ and $(n_\theta+1)/2^n$, only the former approximates θ to precision $2^{-(n+1)}$. By inequalities

$$\frac{2}{\pi}|y| \leq |\sin(y)| \leq |y|, \qquad \forall y \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \tag{3.8}$$

the probability of obtaining $|n_\theta\rangle|\boldsymbol{u}\rangle$ by measurement is

$$\frac{1}{2^{2n}}\left|\sum_{y=0}^{2^n-1} \mathrm{e}^{2\pi \mathrm{i} y\delta(n_\theta)}\right|^2 = \frac{1}{2^{2n}}\left|\frac{\mathrm{e}^{2\pi \mathrm{i} 2^n\delta(n_\theta)} - 1}{\mathrm{e}^{2\pi \mathrm{i}\delta(n_\theta)} - 1}\right|^2 = \frac{1}{2^{2n}}\left|\frac{\sin(2^n\delta(n_\theta)\pi)}{\sin(\delta(n_\theta)\pi)}\right|^2 \geq \frac{4}{\pi^2}. \tag{3.9}$$

Case 3 If $2^{-(n+1)} \leq r_n < 2^{-n}$, then $-\delta(n_\theta + 1) = 2^{-n} - r_n \in (0, 2^{-(n+1)}]$. Similar to Case 2, one gets that between $n_\theta/2^n$ and $(n_\theta+1)/2^n$, only the latter approximates θ to precision $2^{-(n+1)}$. The probability of obtaining $|n_\theta + 1\rangle|\boldsymbol{u}\rangle$ by measurement is $\geq 4/\pi^2$.

By the above analysis, with probability $\geq 4/\pi^2 \approx 0.4$, an approximate of θ to precision $2^{-(n+1)}$ can be obtained. Since the probability is not close to 1, by running the algorithm several times, the probability can be improved to be close to 1.

The following is another method to improve the success probability of the algorithm. Suppose that the precision required is $\varepsilon := 2^{-m}$ for some $0 < m \leq n-2$. (3.5) can be rewritten as

$$\theta = \frac{\theta_1 2^{m-1} + \cdots + \theta_m}{2^m} + r_m =: \frac{m_\theta}{2^m} + r_m, \tag{3.10}$$

where $\theta_i \in \{0,1\}$ and $0 \leq r_m < 2^{-m}$. A similar analysis shows that all elements of $\{(2^{n-m}m_\theta + t)/2^n \mid t = 0, 1, \cdots, 2^{n-m} - 1\}$ are approximates of θ to precision 2^{-m}. It is proved in [12, Subsection 5.2] that the success probability to obtain one of these approximates is at least $1 - \delta := 1 - \frac{1}{2(2^{n-m}-2)}$.

Representing m, n as integer functions in ε, δ would result in

$$n = \left\lceil \log\frac{1}{\varepsilon}\right\rceil + \left\lceil \log\left(2 + \frac{1}{2\delta}\right)\right\rceil = O\left(\log\frac{1}{\varepsilon\delta}\right). \tag{3.11}$$

Proposition 3.1 (see [10]) *Let U be a unitary transformation with implementation complexity $O(T)$, and let $\boldsymbol{u}$ be an eigenvector of U. Then the quantum phase estimation algorithm returns the corresponding eigenvalue in time $O(T/\varepsilon\delta)$ to precision ε, with probability at least $1-\delta$.*

A direct application of the above algorithm is quantum eigenphase decomposition of unitary matrices, which is often taken as another form of quantum phase estimation. Let U be an $M\times M$ unitary matrix, with eigenvalues $\{\mathrm{e}^{2\pi\mathrm{i}\theta_j} \mid j=1,2,\cdots,M\}$ and corresponding unit eigenvectors $\{\boldsymbol{u}_j \mid j=1,2,\cdots,M\}$, where $0\le\theta_j<1$. Let $\boldsymbol{c}$ be an arbitrary nonzero vector of the M-space. Then $\boldsymbol{c}=\sum_{j=1}^M \gamma_j\boldsymbol{u}_j$ for some unknown parameters γ_j, as the $\boldsymbol{u}_j$ form an orthonormal basis of the M-space.

Suppose that U is given, but the $\boldsymbol{u}_j$ are not, nor are the θ_j. In Proposition 3.1, choose n by (3.11), and choose for Algorithm 1 the following initial state:

$$|0\rangle^{\otimes n}|\boldsymbol{c}\rangle=\sum_{j=1}^M\gamma_j|0\rangle^{\otimes n}|\boldsymbol{u}_j\rangle. \tag{3.12}$$

The output of the algorithm is

$$\sum_{j=1}^M\gamma_j|\widetilde{\theta}_j\rangle|\boldsymbol{u}_j\rangle, \tag{3.13}$$

where $\widetilde{\theta}_j$ is an approximate of θ_j for all $1\le j\le M$.

3.2 Application: Order Finding

Let M be a fixed natural number. Given an integer $a\in\mathbb{Z}_M^*$ satisfying $\gcd(a,M)=1$, the smallest positive integer r such that $a^r\equiv 1 \mod M$, is called the *order* of a. The problem of finding the order of an integer in $\mathbb{Z}_M^*$ is called order finding.

Order finding is a problem where quantum phase estimation is a key technique. In the following, we describe the order finding algorithm in Shor's algorithm[9]. Denote $m=\lceil\log M\rceil$. Define unitary transformation

$$U: x\in\mathbb{Z}_{2^m}\mapsto\begin{cases} ax \mod M, & \text{if } 0\le x\le M-1,\\ x, & \text{else.}\end{cases} \tag{3.14}$$

In [12], it was shown that for all $0\le s<r$,

$$|\boldsymbol{u}_s\rangle:=\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\mathrm{e}^{-2\pi\mathrm{i}sk/r}|a^k \mod M\rangle \tag{3.15}$$

is the quantum state of eigenvector $\boldsymbol{u}_s$ of U, and the corresponding eigenvalue is $\mathrm{e}^{2\pi\mathrm{i}s/r}$. Furthermore,

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|\boldsymbol{u}_s\rangle=|0\rangle^{\otimes(m-1)}|1\rangle. \tag{3.16}$$

In Algorithm 1, choose the initial state $|0\rangle^{\otimes n}|\boldsymbol{c}\rangle = |0\rangle^{\otimes n}|0\rangle^{\otimes(m-1)}|1\rangle$. Following the expression on the left side of (3.16), after Step 4 of the algorithm,

$$|\psi_4\rangle = \frac{1}{2^m\sqrt{r}} \sum_{x=0}^{2^m-1} \sum_{s=0}^{r-1} \left[\sum_{y=0}^{2^m-1} \mathrm{e}^{\mathrm{i}2\pi y(\frac{s}{r}-\frac{x}{2^m})} \right] |y\rangle|\boldsymbol{u}_s\rangle. \tag{3.17}$$

According to the analysis of Algorithm 1, by running the algorithm several times, with high probability a value $0 \leq x < 2^m$ can be obtained, such that s/r is a continued fraction approximate of $x/2^m$ for two unknown integers s, r satisfying $0 \leq s < r$. The integer r is what to be after.

To obtain r, first compute the continued fraction expansion of $x/2^m$:

$$\frac{x}{2^m} = c_0 + \cfrac{1}{c_1 + \cfrac{1}{\ddots + \frac{1}{c_m}}} =: [c_0, c_1, \cdots, c_m], \tag{3.18}$$

where $c_0, c_1, \cdots, c_m$ are positive integers. For every $0 \leq k \leq m$, $[c_0, c_1, \cdots, c_k]$ is called the *k-th convergent* of $x/2^m$. Let $[c_0, c_1, \cdots, c_k] = s_k/r_k$ for $0 \leq k \leq m$. Then for all $0 \leq k \leq m$ such that $r_k \leq M$, check if any of $r_k, 2r_k, \cdots, \lceil M/r_k \rceil r_k$ is the order of a. When the procedure terminates, the order r of a must be found. The complexity of the whole procedure for finding r is $O(\operatorname{poly}\log M)$.

3.3 Swap Test

Swap test is a technique to estimate the inner product of two quantum states. In particular, it can be used to estimate the amplitude of a quantum state in a basis state. The key idea in this technique is quantum phase estimation.

Lemma 3.2 (see [52, 225]) *Let* $|\phi\rangle = \sin(\theta)|0\rangle|u\rangle + \cos(\theta)|1\rangle|v\rangle$ *be a quantum state that can be prepared in time* T*, where* $0 \leq \theta < \pi$*, and* $|u\rangle, |v\rangle \in \mathbb{C}^n$*. Then there is a quantum algorithm that computes* $\sin(\theta), \cos(\theta)$ *in time* $O(T/\varepsilon\delta)$ *to precision* ε *with probability at least* $1 - \delta$.

Proof Rotation $\begin{bmatrix} \cos(2\theta) & \sin(2\theta) \\ -\sin(2\theta) & \cos(2\theta) \end{bmatrix}$ in the 2-space spanned by $|0\rangle|u\rangle, |1\rangle|v\rangle$ has the following decomposition in the entire $(2n)$-space $\mathbb{C}^2 \otimes \mathbb{C}^n$:

$$R_{\cos(2\theta),-\sin(2\theta)} = (I - 2|\phi\rangle\langle\phi|)(R_{|0\rangle} \otimes I), \tag{3.19}$$

where $R_{|0\rangle}$ is the reflection with respect to $|0\rangle$ in the 2-space $\mathbb{C}^2$ spanned by $|0\rangle, |1\rangle$. The eigenvalues of $R_{\cos(2\theta),-\sin(2\theta)}$ are $\mathrm{e}^{\pm \mathrm{i}2\theta}$, and the corresponding eigenvectors are

$$|\boldsymbol{w}_\pm\rangle := \frac{1}{\sqrt{2}}(|0\rangle|u\rangle \pm \mathrm{i}|1\rangle|v\rangle). \tag{3.20}$$

Since

$$|\phi\rangle = -\frac{\mathrm{i}}{\sqrt{2}}(\mathrm{e}^{\mathrm{i}\theta}|\boldsymbol{w}_+\rangle - \mathrm{e}^{-\mathrm{i}\theta}|\boldsymbol{w}_-\rangle), \tag{3.21}$$

by Proposition 3.1, when choosing n according to (3.11), and choosing initial state $|0\rangle^n|\boldsymbol{c}\rangle = |0\rangle^n|\phi\rangle$, then by applying Algorithm 1 to $U = R_{\cos(2\theta),-\sin(2\theta)}$, a value $\widetilde{\theta} \in [0, \pi)$ satisfying

$|\widetilde{\theta}-\theta| \leq \varepsilon$ can be obtained, with failure probability at most δ. Then $\sin(\widetilde{\theta}), \cos(\widetilde{\theta})$ can be computed in negligible time, and $|\sin(\widetilde{\theta})-\sin(\theta)| \leq \varepsilon$, $|\cos(\widetilde{\theta})-\cos(\theta)| \leq \varepsilon$. ▮

The proving procedure of the following proposition gives the swap test method.

Proposition 3.3 (see [52, 225]) *Let $|\boldsymbol{x}\rangle, |\boldsymbol{y}\rangle$ be two quantum states that can be prepared simultaneously in time $O(T)$, then $\langle\boldsymbol{x}|\boldsymbol{y}\rangle$ can be estimated to precision ε in time $O(T/\varepsilon)$.*

Proof Consider the quantum state

$$|\phi\rangle = \frac{1}{\sqrt{2}}\Big(|+\rangle|\boldsymbol{x}\rangle + |-\rangle|\boldsymbol{y}\rangle\Big) = \frac{1}{2}\Big(|0\rangle(|\boldsymbol{x}\rangle+|\boldsymbol{y}\rangle) + |1\rangle(|\boldsymbol{x}\rangle-|\boldsymbol{y}\rangle)\Big). \tag{3.22}$$

The probability of obtaining $|0\rangle$ by measuring the first register of $|\phi\rangle$ is $(1+\mathrm{Re}\langle\boldsymbol{x}|\boldsymbol{y}\rangle)/2$, and the probability of obtaining $|1\rangle$ is $(1-\mathrm{Re}\langle\boldsymbol{x}|\boldsymbol{y}\rangle)/2$.

When both $|\boldsymbol{x}\rangle+|\boldsymbol{y}\rangle$ and $|\boldsymbol{x}\rangle-|\boldsymbol{y}\rangle$ are nonzero, set

$$\sin(\theta) = \sqrt{(1+\mathrm{Re}\langle\boldsymbol{x}|\boldsymbol{y}\rangle)/2}, \qquad \cos(\theta) = \sqrt{(1-\mathrm{Re}\langle\boldsymbol{x}|\boldsymbol{y}\rangle)/2}, \tag{3.23}$$

and

$$\boldsymbol{u} = \frac{|\boldsymbol{x}\rangle+|\boldsymbol{y}\rangle}{\||\boldsymbol{x}\rangle+|\boldsymbol{y}\rangle\|}, \qquad \boldsymbol{v} = \frac{|\boldsymbol{x}\rangle-|\boldsymbol{y}\rangle}{\||\boldsymbol{x}\rangle-|\boldsymbol{y}\rangle\|}. \tag{3.24}$$

Then $\theta \in [0, \pi/2]$, and

$$|\phi\rangle = \sin(\theta)|0\rangle|\boldsymbol{u}\rangle + \cos(\theta)|1\rangle|\boldsymbol{v}\rangle. \tag{3.25}$$

By Lemma 3.2, an ε'-approximate $\widetilde{\theta}$ of θ can be obtained in time $O(T/\varepsilon)$, where ε' is determined as follows. By $\mathrm{Re}\langle\boldsymbol{x}|\boldsymbol{y}\rangle = \cos(2\theta)$,

$$\big|\cos(2\widetilde{\theta}) - \mathrm{Re}\langle\boldsymbol{x}|\boldsymbol{y}\rangle\big| \leq 2|\widetilde{\theta}-\theta| \leq 2\varepsilon'. \tag{3.26}$$

Setting $\varepsilon' = \varepsilon/2$ would guarantee the precision ε of approximating $\mathrm{Re}\langle\boldsymbol{x}|\boldsymbol{y}\rangle$ by $\cos(2\widetilde{\theta})$.

By replacing $|\boldsymbol{y}\rangle$ with $\mathrm{i}|\boldsymbol{y}\rangle$ in (3.22), $\mathrm{Im}\langle\boldsymbol{x}|\boldsymbol{y}\rangle$ can be estimated similarly.

When $|\boldsymbol{x}\rangle = |\boldsymbol{y}\rangle$, (3.22) becomes $|\phi\rangle = |0\rangle|\boldsymbol{x}\rangle = \sin(\pi/2)|0\rangle|\boldsymbol{x}\rangle$, so an approximate $\widetilde{\theta} \in [0,\pi)$ of $\theta = \pi/2$ can be obtained by Lemma 3.2. When $|\boldsymbol{x}\rangle = -|\boldsymbol{y}\rangle$, (3.22) becomes $|\phi\rangle = |1\rangle|\boldsymbol{x}\rangle = \cos(0)|1\rangle|\boldsymbol{x}\rangle$, so an approximate $\widetilde{\theta} \in [0,\pi)$ of $\theta = 0$ can be obtained by Lemma 3.2. ▮

One application of swap test is matrix multiplication computing. The multiplication of two $N \times N$ matrices involves the evaluation of N^2 inner products of vector pairs. If the preparation of the quantum states of the rows and columns of the two matrices are efficient, i.e., in time $T = O(\mathrm{poly}\log N)$, then by Proposition 3.3, the matrix multiplication can be computed in time $O(N^2(\mathrm{poly}\log N)/\varepsilon)$ to precision ε.

3.4 Linear Combination of Unitaries (LCU)

Given N nonzero complex numbers $\alpha_0, \cdots, \alpha_{N-1}$, and N unitary operators $U_0, \cdots, U_{N-1}$ that can be implemented in time T in quantum computer, LCU is on implementing linear (but not unitary) operator $L = \sum_{j=0}^{N-1} \alpha_j U_j$.

LCU was first proposed by Long in 2006 as a basic operation of duality quantum computer[226]. Later on, some other types of quantum algorithms to realize LCU were proposed[118, 222]. The importance of LCU algorithms was recognized only in recent years, in the study of Hamiltonian simulation[105–107, 114, 227], quantum linear solver[115], quantum walk[114], quantum state preparation[118], quantum gradient descent[228, 229], quantum Newton's method[229], etc. In the following, we describe the two LCU algorithms in [118, 222] respectively, and show the application of LCU in the preparation of weighted superposition of quantum states.

To implement L, it suffices to construct a quantum algorithm to prepare $L|\phi\rangle$ for any given initial state $|\phi\rangle$. In the following algorithm[118], $t = 1/\max_j |\alpha_j|$ is assumed to be given. Alternatively, t can be replaced by a relatively tight (but known) lower bound of the $1/|\alpha_j|$.

For any $0 \le j < N$, $R_j := R_{t\alpha_j, \sqrt{1-t^2|\alpha_j|^2}}$ is the complex rotation in the 2-space spanned by $|0\rangle, |1\rangle$ such that $R_j|0\rangle = t\alpha_j|0\rangle + \sqrt{1-t^2|\alpha_j|^2}|1\rangle$.

Algorithm 2 Linear combinations of unitaries (version 1)[118]

```
1: Prepare the initial state $|\psi_1\rangle = |\phi\rangle|\varpi_N\rangle|0\rangle|0\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|\phi\rangle|j\rangle|0\rangle|0\rangle$.
2: Use the second register $|j\rangle$ of $|\psi_1\rangle$ as the control register, generate state values $|\alpha_j\rangle$ in the third register of $|\psi_1\rangle$ for all $0 \le j < N$ (control transformation). Result:
       $$|\psi_2\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|\phi\rangle|j\rangle|\alpha_j\rangle|0\rangle. \quad (3.27)$$
3: Apply control transformation $\sum_{j=0}^{N-1} U_j \otimes |j\rangle\langle j| \otimes I \otimes R_j$ to $|\psi_2\rangle$. Result:
       $$|\psi_3\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} U_j|\phi\rangle|j\rangle|\alpha_j\rangle\left[t\alpha_j|0\rangle + \sqrt{1-t^2|\alpha_j|^2}|1\rangle\right]. \quad (3.28)$$
4: Eliminate from the third register of $|\psi_3\rangle$ the state values $|\alpha_j\rangle$ for all $0 \le j < N$ (inverse of the unitary transformation in Step 2). Result:
       $$|\psi_4\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} U_j|\phi\rangle|j\rangle|0\rangle\left[t\alpha_j|0\rangle + \sqrt{1-t^2|\alpha_j|^2}|1\rangle\right]. \quad (3.29)$$
5: Apply Hadamard transform to the third register $|j\rangle$ of $|\psi_4\rangle$. Result:
       $$|\psi_5\rangle = \frac{t}{N}\sum_{j=0}^{N-1}\alpha_j U_j|\phi\rangle|0,0,0\rangle + (\cdots)|0,0,0\rangle^\perp = \frac{t}{N}L|\phi\rangle|0,0,0\rangle + (\cdots)|0,0,0\rangle^\perp. \quad (3.30)$$
6: Perform measurement to $|\psi_5\rangle$ at the last three registers.
```

In the last step, the probability to get $L|\phi\rangle$ by measurement is $\|L|\phi\rangle\|^2 t^2/N^2$. In Step 1 to Step 5, the running time is respectively $O(\log N), O(1), O(T), O(1), O(\log N)$. By quantum amplitude amplification[52], the time complexity of the algorithm is

$$O\Big((T + \log N)N \max_j |\alpha_j| \Big/ \|L|\phi\rangle\|\Big). \tag{3.31}$$

One application of LCU is quantum state preparation. Given a vector $\boldsymbol{x} = (\alpha_0, \cdots, \alpha_{N-1})$, preparing unnormalized quantum state $\|\boldsymbol{x}\|\,|\boldsymbol{x}\rangle = \sum_{j=0}^{N-1} \alpha_j |j\rangle$ is called *quantum state preparation*, or the *input problem*. This problem can be a serious bottleneck[99, 113, 115, 124, 127, 129, 228]. In the extreme case, the cost of quantum state preparation can dominate the cost of the whole quantum algorithm[214, 216]. Quantum algorithms to solve the input problem can be found in [118, 176, 230, 231].

Proposition 3.4 (see [118]) *For any vector $\boldsymbol{x} = (\alpha_0, \cdots, \alpha_{N-1})$, $\|\boldsymbol{x}\|\,|\boldsymbol{x}\rangle$ can be prepared in time $O(\kappa \log N)$, where $\kappa = \max_j |\alpha_j| / \min_{j,\alpha_j \neq 0} |\alpha_j|$ is the condition number of vector $\boldsymbol{x}$.*

Proof Only the nonzero entries of $\boldsymbol{x}$ function in $\|\boldsymbol{x}\|\,|\boldsymbol{x}\rangle$. If there is any component of $\boldsymbol{x}$ with zero value, then only the components with nonzero values are visited in the LCU algorithm, under the assumption that in $\boldsymbol{x}$, the components with zero value are given beforehand. Without loss of generality, assume $\alpha_j \neq 0$ for all $0 \leq j < N$.

In Algorithm 2, by choosing $|\phi\rangle = |0\rangle$, and $U_j = I$ for all j, the result of Step 4 is (after omitting common constant state values):

$$\frac{t}{\sqrt{N}} \sum_{j=0}^{N-1} \alpha_j |j\rangle|0\rangle + |0\rangle^{\perp} = \frac{t\|\boldsymbol{x}\|}{\sqrt{N}} |\boldsymbol{x}\rangle|0\rangle + |0\rangle^{\perp}. \tag{3.32}$$

By measuring (3.32) at the last register, $\|\boldsymbol{x}\||\boldsymbol{x}\rangle$ is obtained with probability $t^2\|\boldsymbol{x}\|^2/N \geq 1/\kappa^2$. As $T = 1$ in (3.31), the complexity to prepare $\|\boldsymbol{x}\|\,|\boldsymbol{x}\rangle$ is $O(\kappa \log N)$. ∎

A direct corollary of Proposition 3.4 is that the quantum state of a relatively uniformly distributed vector can be prepared efficiently in quantum computer.

Next we introduce the LCU algorithm proposed in [222]. In the following,

$$\alpha := \sum_{j=0}^{N-1} |\alpha_j|, \qquad \mathrm{e}^{\mathrm{i}\theta_j} := \frac{\alpha_j}{|\alpha_j|}, \tag{3.33}$$

and V is a unitary transformation satisfying $V|0\rangle = \frac{1}{\sqrt{\alpha}} \sum_{j=0}^{N-1} \sqrt{|\alpha_j|}\,|j\rangle$.

Algorithm 3 Linear combination of unitaries (version 2)[222]

1: Prepare the initial state $|\psi_1\rangle = |\phi\rangle|0\rangle$.

2: Apply $I \otimes V$ to $|\psi_1\rangle$. Result:

$$|\psi_2\rangle = \frac{1}{\sqrt{\alpha}} \sum_{j=0}^{N-1} \sqrt{|\alpha_j|}\,|\phi\rangle|j\rangle. \tag{3.34}$$

3: Use control transformation to insert the phase $\mathrm{e}^{\mathrm{i}\theta_j}$ into the coefficient of $|j\rangle$, and then apply U_j to $|\phi\rangle$, with the last register $|j\rangle$ of $|\psi_2\rangle$ as the control register. Result:

$$|\psi_3\rangle = \frac{1}{\sqrt{\alpha}} \sum_{j=0}^{N-1} \sqrt{|\alpha_j|\mathrm{e}^{\mathrm{i}\theta_j}}\,(U_j|\phi\rangle) \otimes |j\rangle. \tag{3.35}$$

```
4: Apply $I \otimes V^{-1}$ to $|\psi_3\rangle$. Result:
```

$$|\psi_4\rangle = \frac{1}{\alpha}\sum_{j=0}^{N-1}\alpha_j U_j|\phi\rangle|0\rangle + (\cdots)|0\rangle^{\perp} = \frac{1}{\alpha}L|\phi\rangle|0\rangle + (\cdots)|0\rangle^{\perp}. \tag{3.36}$$

```
5: Perform measurement to $|\psi_4\rangle$ at the last register.
```

The complexity of the above algorithm is $O((T + C_V)\alpha/\|L|\phi\rangle\|)$, where $O(C_V)$ is the complexity to implement V in quantum computer. Since $\alpha \le N \max_j |\alpha_j|$, if $O(C_V) = O(\log N)$, then Algorithm 3 is more efficient than Algorithm 2.

4 Quantum Linear Solvers with Applications in Machine Learning

Solving linear equations is a fundamental problem in many disciplines of science and engineering. A typical example is machine learning, which involves manipulating and classifying a large number of vectors in a high dimensional space[232, 233]. Many machine learning tasks are related to optimization problems that can be reduced to linear system solving[234].

For linear system solving, one of the best classical algorithms – the conjugate gradient method[117, 235, 236], has complexity $O(Ms\sqrt{\kappa}\log 1/\varepsilon)$, where M is the maximum between the number of equations and the number of variables, s, κ are the sparsity and the condition number of the coefficient matrix respectively, and ε is the precision. In comparison, the quantum algorithms in [99, 115, 116, 118, 119] achieve exponential speedup in parameter M.

4.1 The First Quantum Linear Solver: HHL Algorithm

Consider a linear system $A\boldsymbol{x} = \boldsymbol{b}$, where A is an invertible $M \times M$ Hermitian matrix with condition number κ. If this is not the case, then consider the following equivalent linear system:

$$\begin{bmatrix} 0 & A \\ A^{\dagger} & 0 \end{bmatrix}\begin{bmatrix} 0 \\ \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ 0 \end{bmatrix}. \tag{4.1}$$

Since A is Hermitian, $\mathrm{e}^{-\mathrm{i}tA}$ is unitary, and can be efficiently implemented in quantum computer when A is sparse. Assume that realizing $\mathrm{e}^{-\mathrm{i}tA}$ in quantum computer has complexity $O(t\log M)$. Further assume that quantum state $|\boldsymbol{b}\rangle$ can be prepared efficiently.

Let $A = \sum_{j=1}^{M}\delta_j\sigma_j|\boldsymbol{u}_j\rangle\langle\boldsymbol{u}_j|$ be an SVD of A, where the $\sigma_j > 0$, and for $\delta_j \in \{1, -1\}$, the $\delta_j\sigma_j$ are eigenvalues of A. Then $|\boldsymbol{b}\rangle = \sum_{j=1}^{M}\gamma_j|\boldsymbol{u}_j\rangle$ for some unknown parameters γ_j such that $\sum_{j=1}^{M}|\gamma_j|^2 = 1$.

By a suitable re-scaling of the original linear system, it can be assumed that $\|A\|_{\mathrm{sp}} = 1$. Then $\|A^{-1}\|_{\mathrm{sp}} = \kappa > 1$, and

$$\kappa^{-1} \le \sigma_j \le 1, \quad \forall 1 \le j \le M. \tag{4.2}$$

Assume that a lower bound $K = \Theta(1/\kappa)$ of the σ_j is given. Then $K \le \kappa^{-1} < 1$, and for all $1 \le j \le M$,

$$K \le \sigma_j \le 1, \qquad K \le K\sigma_j^{-1} \le 1. \tag{4.3}$$

The quantum state $|\boldsymbol{x}\rangle$ of a solution $\boldsymbol{x}$ to $A\boldsymbol{x} = \boldsymbol{b}$ is proportional to

$$A^{-1}|\boldsymbol{b}\rangle = \sum_{j=1}^{M} \gamma_j \delta_j \sigma_j^{-1} |\boldsymbol{u}_j\rangle. \tag{4.4}$$

The following HHL algorithm is based on this observation.

Step 1 By quantum eigenphase decomposition (3.13), when choosing $|0\rangle|\boldsymbol{b}\rangle|0\rangle$ as the initial state, the following quantum state can be obtained in time $O((\log M)/\varepsilon')$:

$$\sum_{j=1}^{M} \gamma_j |\widetilde{\delta_j\sigma_j}\rangle |\boldsymbol{u}_j\rangle |0\rangle, \tag{4.5}$$

where $|\widetilde{\delta_j\sigma_j}| \in [K, 1]$ and $|\widetilde{\delta_j\sigma_j} - \delta_j\sigma_j| \le \varepsilon'$ for all $1 \le j \le M$. The precision ε' will be made explicit in the analysis of the algorithm.

Step 2 In order to put $\widetilde{\delta_j\sigma_j}^{-1}$ into the coefficient of (4.5) for $1 \le j \le M$ so that (4.4) can be obtained, apply the 2-dimensional rotation

$$R_{K\widetilde{\delta_j\sigma_j}^{-1}, \sqrt{1-K^2\widetilde{\delta_j\sigma_j}^{-2}}} = \begin{bmatrix} K\widetilde{\delta_j\sigma_j}^{-1} & -\sqrt{1-K^2\widetilde{\delta_j\sigma_j}^{-2}} \\ \sqrt{1-K^2\widetilde{\delta_j\sigma_j}^{-2}} & K\widetilde{\delta_j\sigma_j}^{-1} \end{bmatrix}, \tag{4.6}$$

which is controlled by the first register $|\widetilde{\delta_j\sigma_j}\rangle$ of (4.5), to the third register. Then (4.5) is changed into

$$\sum_{j=1}^{M} \gamma_j |\widetilde{\delta_j\sigma_j}\rangle |\boldsymbol{u}_j\rangle \left[K\widetilde{\delta_j\sigma_j}^{-1}|0\rangle + \sqrt{1-K^2\widetilde{\delta_j\sigma_j}^{-2}}|1\rangle \right]. \tag{4.7}$$

Step 3 Undo the quantum eigenphase decomposition of Step 1 to eliminate from the first register of (4.7) the state values $|\widetilde{\delta_j\sigma_j}\rangle$ for all $1 \le j \le M$. The result from (4.7) is

$$\sum_{j=1}^{M} \gamma_j |0\rangle |\boldsymbol{u}_j\rangle \left[K\widetilde{\delta_j\sigma_j}^{-1}|0\rangle + \sqrt{1-K^2\widetilde{\delta_j\sigma_j}^{-2}}|1\rangle \right] = |0\rangle \big(Kt|\boldsymbol{x}\rangle|0\rangle + (\cdots)|0\rangle^{\perp} \big), \tag{4.8}$$

where

$$t|\boldsymbol{x}\rangle := \sum_{j=1}^{M} \gamma_j \widetilde{\delta_j\sigma_j}^{-1} |\boldsymbol{u}_j\rangle. \tag{4.9}$$

Step 4 By measuring (4.8) at the first and the last registers, the quantum state $|\boldsymbol{x}\rangle$ can be obtained with probability

$$K^2 t^2 = \sum_{j=1}^{M} K^2 \frac{|\gamma_j|^2}{\widetilde{\delta_j\sigma_j}^{2}} \ge K^2. \tag{4.10}$$

To improve the success probability by amplitude amplification, the algorithm needs to run $O(1/K)$ times.

By

$$\| t|\boldsymbol{x}\rangle - A^{-1}|\boldsymbol{b}\rangle \| = \left\| \sum_{j=1}^{M} \gamma_j \big(\widetilde{\delta_j\sigma_j}^{-1} - \delta_j\sigma_j^{-1} \big) |\boldsymbol{u}_j\rangle \right\| \le \max_j |\widetilde{\delta_j\sigma_j}^{-1} - \delta_j\sigma_j^{-1}|, \tag{4.11}$$

to control the precision to ε in the sense of (2.28), i.e., $\big\|t|\boldsymbol{x}\rangle - A^{-1}|\boldsymbol{b}\rangle\big\| \leq \varepsilon\kappa$, it suffices to set

$$\max_j |\widetilde{\delta_j\sigma_j}^{-1} - \delta_j\sigma_j^{-1}| \leq \varepsilon\kappa. \tag{4.12}$$

By

$$|\widetilde{\delta_j\sigma_j}^{-1} - \delta_j\sigma_j^{-1}| = \frac{|\delta_j\sigma_j - \widetilde{\delta_j\sigma_j}|}{|\delta_j\sigma_j\,\widetilde{\delta_j\sigma_j}|} \leq \frac{\varepsilon'}{K^2}, \tag{4.13}$$

It suffice to choose $\varepsilon' = \varepsilon\kappa K^2 \leq \varepsilon/\kappa$.

In the four steps of the algorithm, Step 2 has constant complexity, Step 1 and Step 3 have the same complexity $O((\log M)/\varepsilon')$, or equivalently, $O(\kappa(\log M)/\varepsilon)$. So the whole algorithm has complexity $O(\kappa^2(\log M)/\varepsilon)$.

In 2012, Ambainis[116] made an improvement over HHL algorithm and reduced factor κ^2 to $\kappa\log^3\kappa$ in the complexity. In 2017, Childs, et al.[115] made a further improvement and reduced factor $1/\varepsilon$ to $\log(1/\varepsilon)$.

Remark 4.1 The following properties of HHL algorithm can be found in [237–239]:

1) When $A\boldsymbol{x} = \boldsymbol{b}$ has more than one solution, HHL algorithm only returns one solution, which is the least-square solution. More precisely, the algorithm solves $A^\dagger A\boldsymbol{x} = A^\dagger\boldsymbol{b}$ instead of $A\boldsymbol{x} = \boldsymbol{b}$.

2) HHL algorithm needs efficient preparation of the quantum state $|\boldsymbol{b}\rangle$ of $\boldsymbol{b}$.

3) The output of HHL algorithm is a quantum state $|\boldsymbol{x}\rangle$. Obtaining the classical solution $\boldsymbol{x}$ up to normalization from $|\boldsymbol{x}\rangle$ takes at least $O(M)$ steps, which sacrifices the exponential speedup of HHL algorithm.

4) The inner product of the solution state $|\boldsymbol{x}\rangle$ with any other quantum state can be estimated by swap test. This is enough for many applications in machine learning.

5) HHL algorithm requires either A to be invertible, or $\boldsymbol{b}$ to lie in the well-conditioned region of A: If singular value $\sigma_j = 0$ for some $1 \leq j \leq M$, then in $\boldsymbol{b}$, the component γ_j with respect to the corresponding eigenvector $\boldsymbol{u}_j$ is zero.

4.2 Quantum Linear Solving by Singular Value Estimation (SVE)

The efficiency of HHL algorithm relies on efficient implementation of Hamiltonian $\mathrm{e}^{-\mathrm{i}At}$. However, Hamiltonian simulation is efficient only in some special cases. In 2017, Kerenidis and Prakash[113] introduced a tree structure similar to QRAM[240, 241] to store a matrix in quantum computer efficiently, and in this way discovered a fast algorithm for quantum SVE. Nowadays, SVE has many important applications in machine learning[109, 113, 119, 228, 242] and Hamiltonian simulation[109, 114]. In 2018, Wossnig, et al.[119] proposed a new quantum algorithm to solve general dense linear systems based on SVE.

Below we first introduce Kerenidis, et al.'s quantum SVE, then introduce Wossnig, et al.'s quantum linear solver.

Let $A = (a_{ij})$ be an $M \times M$ matrix none of whose rows is composed of zeros. Denote the i-th row of A by A_{i-1}^{T}, where $1 \leq i \leq M$. Then $\|A_i\| \neq 0$ for $0 \leq i < M$. Denote

$$A_{\|} := (\|A_0\|, \|A_1\|, \cdots, \|A_{M-1}\|)^{\mathrm{T}}. \tag{4.14}$$

Then

$$|A_i\rangle = \frac{1}{\|A_i\|}\sum_{j=0}^{M-1} a_{ij}|j\rangle, \qquad |A_{\|}\rangle = \frac{1}{\|A\|}\sum_{i=0}^{M-1} \|A_i\|\,|i\rangle. \tag{4.15}$$

Define two linear operators $L_{\mathcal{M}}$ and $L_{\mathcal{N}}$ from $\mathbb{C}^M$ to $\mathbb{C}^M \otimes \mathbb{C}^M$ as following: For all $i, j \in \{0, 1, \cdots, M-1\}$,

$$L_{\mathcal{M}}:\ |i\rangle \mapsto |i\rangle|A_i\rangle, \qquad L_{\mathcal{N}}:\ |j\rangle \mapsto |A_{\|}\rangle|j\rangle. \tag{4.16}$$

Then $L_{\mathcal{M}}^\dagger L_{\mathcal{N}} = \|A\|^{-1}A$. Furthermore, $L_{\mathcal{M}}^\dagger L_{\mathcal{M}} = L_{\mathcal{N}}^\dagger L_{\mathcal{N}} = I_M$, the latter being the $M \times M$ identity matrix.

Denote by $\mathcal{V}_{\mathcal{M}}, \mathcal{V}_{\mathcal{N}}$ the M-spaces spanned respectively by the images of $L_{\mathcal{M}}, L_{\mathcal{N}}$ in $\mathbb{C}^M \otimes \mathbb{C}^M$. Let $U_{\mathcal{M}}, U_{\mathcal{N}}$ be two unitary operators in $\mathbb{C}^M \otimes \mathbb{C}^M$ such that for all $i, j \in \{0, 1, \cdots, M-1\}$,

$$U_{\mathcal{M}} : |i\rangle|0\rangle \mapsto |i\rangle|A_i\rangle, \qquad U_{\mathcal{N}} : |0\rangle|j\rangle \mapsto |A_{\|}\rangle|j\rangle. \tag{4.17}$$

In [113], it was shown that there exist two unitary operators with the above property that can be performed in time $O(\operatorname{poly}\log M)$.

In some sense, $U_{\mathcal{M}}$ is a unitary realization of the normalized row vectors of matrix A, with the first register $|i\rangle$ labeling the rows, and the second register $|A_i\rangle$ storing the normalized row vectors in quantum form. On the other hand, $U_{\mathcal{N}}$ takes down the "normalized" norm information of the row vectors in the first register, leaving the second register for free.

Let $R_{\mathcal{V}_{\mathcal{M}}}, R_{\mathcal{V}_{\mathcal{N}}}$ be the reflections in $\mathbb{C}^M \otimes \mathbb{C}^M$ with respect to $\mathcal{V}_{\mathcal{M}}, \mathcal{V}_{\mathcal{N}}$ respectively. By direct computation,

$$\begin{aligned} R_{\mathcal{V}_{\mathcal{M}}} &= 2L_{\mathcal{M}}L_{\mathcal{M}}^\dagger - I_{M^2} = U_{\mathcal{M}}\left(2\sum_{i=0}^{M-1}|i,0\rangle\langle i,0| - I_{M^2}\right)U_{\mathcal{M}}^\dagger, \\ R_{\mathcal{V}_{\mathcal{N}}} &= 2L_{\mathcal{N}}L_{\mathcal{N}}^\dagger - I_{M^2} = U_{\mathcal{N}}\left(2\sum_{j=0}^{M-1}|0,j\rangle\langle 0,j| - I_{M^2}\right)U_{\mathcal{N}}^\dagger. \end{aligned} \tag{4.18}$$

So $R_{\mathcal{V}_{\mathcal{M}}}, R_{\mathcal{V}_{\mathcal{N}}}$ can be efficiently implemented in quantum computer. Denote

$$W = R_{\mathcal{V}_{\mathcal{M}}}R_{\mathcal{V}_{\mathcal{N}}}. \tag{4.19}$$

Then unitary operator W can be implemented efficiently.

Let $A = \sum_{i=0}^{M-1}\sigma_i \boldsymbol{u}_i\boldsymbol{v}_i^\dagger$ be an (unknown) SVD of matrix A. Then

$$\max_i \sigma_i = \|A\|_{\text{sp}} \le \|A\|. \tag{4.20}$$

For any fixed $0 \le k < M$, denote the 2-space spanned by $L_{\mathcal{M}}|\boldsymbol{u}_k\rangle, L_{\mathcal{N}}|\boldsymbol{v}_k\rangle$ in $\mathbb{C}^M \otimes \mathbb{C}^M$ as $\mathcal{V}_k^2$. By

$$\begin{aligned} WL_{\mathcal{N}}|\boldsymbol{v}_k\rangle &= \frac{2\sigma_k}{\|A\|}L_{\mathcal{M}}|\boldsymbol{u}_k\rangle - L_{\mathcal{N}}|\boldsymbol{v}_k\rangle, \\ WL_{\mathcal{M}}|\boldsymbol{u}_k\rangle &= \left(\frac{4\sigma_k^2}{\|A\|^2} - 1\right)L_{\mathcal{M}}|\boldsymbol{u}_k\rangle - \frac{2\sigma_k}{\|A\|}L_{\mathcal{N}}|\boldsymbol{v}_k\rangle, \end{aligned} \tag{4.21}$$

$\mathcal{V}_k^2$ is invariant under W.

An orthonormal basis of the 2-space $\mathcal{V}_k^2$ is

$$\begin{aligned}|\boldsymbol{e}_{k1}\rangle &:= L_{\mathcal{M}}|\boldsymbol{u}_k\rangle,\\ |\boldsymbol{e}_{k2}\rangle &:= \frac{L_{\mathcal{N}}|\boldsymbol{v}_k\rangle - \langle\boldsymbol{e}_{k1}|L_{\mathcal{N}}|\boldsymbol{v}_k\rangle|\boldsymbol{e}_{k1}\rangle}{\|L_{\mathcal{N}}|\boldsymbol{v}_k\rangle - \langle\boldsymbol{e}_{k1}|L_{\mathcal{N}}|\boldsymbol{v}_k\rangle|\boldsymbol{e}_{k1}\rangle\|} = \frac{L_{\mathcal{N}}|\boldsymbol{v}_k\rangle - \sigma_k\|A\|^{-1}|\boldsymbol{e}_{k1}\rangle}{\sqrt{1-\sigma_k^2\|A\|^{-2}}}.\end{aligned} \tag{4.22}$$

Substituting (4.22) into (4.21), we get

$$\begin{aligned}W|\boldsymbol{e}_{k1}\rangle &= \left(\frac{2\sigma_k^2}{\|A\|^2}-1\right)|\boldsymbol{e}_{k1}\rangle - \frac{2\sigma_k}{\|A\|}\sqrt{1-\frac{\sigma_k^2}{\|A\|^2}}|\boldsymbol{e}_{k2}\rangle,\\ W|\boldsymbol{e}_{k2}\rangle &= \frac{2\sigma_k}{\|A\|}\sqrt{1-\frac{\sigma_k^2}{\|A\|^2}}|\boldsymbol{e}_{k1}\rangle + \left(\frac{2\sigma_k^2}{\|A\|^2}-1\right)|\boldsymbol{e}_{k2}\rangle.\end{aligned} \tag{4.23}$$

So W is a 2-dimensional real rotation when restricted to $\mathcal{V}_k^2$.

Denote the eigenvalues of the restriction of W on $\mathcal{V}_k^2$ by $\exp(\pm\mathrm{i}\theta_k)$, where $\theta_k \in [0,\pi)$. Then the corresponding eigenvectors are

$$|\boldsymbol{w}_k^{\pm}\rangle := \frac{1}{\sqrt{2}}(|\boldsymbol{e}_{k1}\rangle \pm \mathrm{i}|\boldsymbol{e}_{k2}\rangle). \tag{4.24}$$

Furthermore,

$$\cos\theta_k = \frac{2\sigma_k^2}{\|A\|^2}-1, \quad \text{i.e.,} \quad \sigma_k = \|A\|\cos\left(\frac{\theta_k}{2}\right). \tag{4.25}$$

Simple calculation gives

$$\begin{aligned}\mathcal{M}|\boldsymbol{u}_k\rangle &= \frac{1}{\sqrt{2}}(|\boldsymbol{w}_k^+\rangle + |\boldsymbol{w}_k^-\rangle),\\ \mathcal{N}|\boldsymbol{v}_k\rangle &= \frac{1}{\sqrt{2}}(\mathrm{e}^{\mathrm{i}\theta_k/2}|\boldsymbol{w}_k^+\rangle + \mathrm{e}^{-\mathrm{i}\theta_k/2}|\boldsymbol{w}_k^-\rangle).\end{aligned} \tag{4.26}$$

With the above notations, given an $M\times M$ matrix A together with its Frobenius norm $\|A\|$, the algorithm given in [113] realizing the SVD of A in the form

$$\sum_{k=0}^{M-1}\alpha_k|\boldsymbol{v}_k\rangle \mapsto \sum_{k=0}^{M-1}\alpha_k|\boldsymbol{u}_k\rangle|\widetilde{\sigma}_k\rangle + O(\varepsilon), \tag{4.27}$$

for all $\alpha_k \in \mathbb{C}$ satisfying $\sum_{k=0}^{M-1}|\alpha_k|^2 = 1$, such that $|\widetilde{\sigma}_k - \sigma_k| \le \varepsilon\|A\|$ for all k, can be described as follows:

Algorithm 4 Quantum singular value estimation (SVE)[113]

```
1: (Preparation of an initial state for the eigenphase decomposition of $W$)
   Choose an initial state $|\psi_0\rangle \in \mathbb{C}^M$, so that it can be formally written as $\sum_{k=0}^{M-1}\alpha_k|\boldsymbol{v}_k\rangle$,
   where the (unknown) $\alpha_k \in \mathbb{C}$. Apply $U_{\mathcal{N}}$ to $|\psi_0\rangle$. Result:
```

$$|\psi_1\rangle = \sum_{k=0}^{M-1}\alpha_k L_{\mathcal{N}}|\boldsymbol{v}_k\rangle = \sum_{k=0}^{M-1}\frac{1}{\sqrt{2}}\alpha_k(\mathrm{e}^{\mathrm{i}\theta_k/2}|\boldsymbol{w}_k^+\rangle + \mathrm{e}^{-\mathrm{i}\theta_k/2}|\boldsymbol{w}_k^-\rangle). \tag{4.28}$$

2: Perform quantum eigenphase decomposition to W, with $|\psi_1\rangle$ as the initial state. The result in the form of (3.13) is:

$$|\psi_2\rangle = \sum_{k=0}^{M-1} \frac{1}{\sqrt{2}}\alpha_k\big(\mathrm{e}^{\mathrm{i}\theta_k/2}|\boldsymbol{w}_k^+\rangle|\widetilde{\theta}_k\rangle + \mathrm{e}^{-\mathrm{i}\theta_k/2}|\boldsymbol{w}_k^-\rangle|-\widetilde{\theta}_k\rangle\big), \tag{4.29}$$

where $\widetilde{\theta}_k$ is a (2ε)-approximate of θ_k for $0 \le k < M$.
By (4.25),

$$\sigma_k = \|A\|\cos(\pm\theta_k/2), \qquad \widetilde{\sigma}_k := \|A\|\cos(\pm\widetilde{\theta}_k/2), \tag{4.30}$$

and $|\widetilde{\sigma}_k - \sigma_k| \le \varepsilon\|A\|$.

3: Use the second register $|\pm\widetilde{\theta}_k\rangle$ of $|\psi_2\rangle$ as the control register, save values $\widetilde{\sigma}_k$ for $0 \le k < M$ in a "new" register (an omitted register having constant state value before). Result:

$$|\psi_3\rangle = \sum_{k=0}^{M-1} \frac{1}{\sqrt{2}}\alpha_k\big(\mathrm{e}^{\mathrm{i}\theta_k/2}|\boldsymbol{w}_k^+\rangle|\widetilde{\theta}_k\rangle + \mathrm{e}^{-\mathrm{i}\theta_k/2}|\boldsymbol{w}_k^-\rangle|-\widetilde{\theta}_k\rangle\big)|\widetilde{\sigma}_k\rangle. \tag{4.31}$$

4: Use the second register $|\pm\widetilde{\theta}_k\rangle$ of $|\psi_3\rangle$ as the control register, put phase factor $\mathrm{e}^{\mp\mathrm{i}\widetilde{\theta}_k/2}$ in each term. The purpose is to eliminate "approximately" the original phase factors $\mathrm{e}^{\pm\mathrm{i}\theta_k/2}$ for $0 \le k < M$. Result:

$$\begin{aligned}|\psi_4\rangle &= \sum_{k=0}^{M-1} \frac{1}{\sqrt{2}}\alpha_k\big(\mathrm{e}^{\mathrm{i}\frac{\theta_k-\widetilde{\theta}_k}{2}}|\boldsymbol{w}_k^+\rangle|\widetilde{\theta}_k\rangle + \mathrm{e}^{-\mathrm{i}\frac{\theta_k-\widetilde{\theta}_k}{2}}|\boldsymbol{w}_k^-\rangle|-\widetilde{\theta}_k\rangle\big)|\widetilde{\sigma}_k\rangle \\ &= \sum_{k=0}^{M-1} \frac{1}{\sqrt{2}}\alpha_k\big(|\boldsymbol{w}_k^+\rangle|\widetilde{\theta}_k\rangle + |\boldsymbol{w}_k^-\rangle|-\widetilde{\theta}_k\rangle\big)|\widetilde{\sigma}_k\rangle + O(\varepsilon). \end{aligned} \tag{4.32}$$

In the second equality of (4.32),

$$\begin{aligned} &\Big\| |\psi_4\rangle - \sum_{k=0}^{M-1} \frac{1}{\sqrt{2}}\alpha_k\big(|\boldsymbol{w}_k^+\rangle|\widetilde{\theta}_k\rangle + |\boldsymbol{w}_k^-\rangle|-\widetilde{\theta}_k\rangle\big)|\widetilde{\sigma}_k\rangle\Big\|^2 \\ &\le \sum_{k=0}^{M-1} \frac{1}{2}|\alpha_k|^2\big(|\mathrm{e}^{\mathrm{i}\frac{\theta_k-\widetilde{\theta}_k}{2}} - 1|^2 + |\mathrm{e}^{-\mathrm{i}\frac{\theta_k-\widetilde{\theta}_k}{2}} - 1|^2\big) \\ &\le \sum_{k=0}^{M-1} \frac{1}{4}|\alpha_k|^2|\theta_k - \widetilde{\theta}_k|^2 \ \le\ \varepsilon^2. \end{aligned} \tag{4.33}$$

5: Undo the quantum eigenphase decomposition of Step 2. The purpose is to eliminate state values $|\pm\widetilde{\theta}_k\rangle$ for $0 \le k < M$ from the second register of $|\psi_4\rangle$. Result (after omitting the second register):

$$|\psi_5\rangle = \sum_{k=0}^{M-1} \frac{1}{\sqrt{2}}\alpha_k(|\boldsymbol{w}_k^+\rangle + |\boldsymbol{w}_k^-\rangle)|\widetilde{\sigma}_k\rangle + O(\varepsilon) = \sum_{k=0}^{M-1} \alpha_k L_{\mathcal{M}}|\boldsymbol{u}_k\rangle|\widetilde{\sigma}_k\rangle + O(\varepsilon). \tag{4.34}$$

6: Apply $U_{\mathcal{M}}^{-1}$. Result (after omitting a register of constant state value):

$$|\psi_6\rangle = \sum_{k=0}^{M-1} \alpha_k |\boldsymbol{u}_k\rangle|\widetilde{\sigma}_k\rangle + O(\varepsilon). \tag{4.35}$$

To be more accurate, in Step 2 of the above algorithm, $-\widetilde{\theta}_k$ should be replaced by an $\varepsilon\|A\|$-approximate $\widetilde{-\theta_k}$ of $-\theta_k$, and in general, $\widetilde{-\theta_k} \neq -\widetilde{\theta}_k$. Accordingly, $\widetilde{\sigma}'_k := \|A\| \cos(\widetilde{-\theta_k}/2)$ should be introduced to distinguish with $\widetilde{\sigma}_k$. These minute distinctions contribute a total of $O(\varepsilon)$ to the final result.

The complexity of Algorithm 4 is $O(\varepsilon^{-1} \operatorname{poly}\log(M))$. Similarly, linear map $\sum_{k=0}^{M-1} \alpha_k|\boldsymbol{u}_k\rangle \mapsto \sum_{k=0}^{M-1} \alpha_k|\boldsymbol{v}_k\rangle|\widetilde{\sigma}_k\rangle$ can be realized with the same complexity.

The following quantum linear solver of Wossnig, et al.[119] is based on quantum SVE. Its main difference in applicability from HHL algorithm, is that the algorithm is applicable to any dense linear system with no assumption on Hamiltonian simulation. The algorithm has complexity $O(\varepsilon^{-1}\kappa^2\|A\| \operatorname{poly}\log(M))$. Remark 4.1 also holds for this algorithm.

Consider linear system $A\boldsymbol{x} = \boldsymbol{b}$, where A is an $M \times M$ positive-definite Hermitian matrix (such a matrix is selected for the purpose of illustration). Its SVD is of the form $A = \sum_{k=0}^{M-1} \sigma_k \boldsymbol{u}_k \boldsymbol{u}_k^\dagger$, where the $\sigma_k > 0$.

Step 1 Choose an initial state formally in the form of

$$|\boldsymbol{b}\rangle = \sum_{k=0}^{M-1} \alpha_k |\boldsymbol{u}_k\rangle, \tag{4.36}$$

where the $\alpha_k \in \mathbb{C}$ satisfy $\sum_{k=1}^{M-1} |\alpha_k|^2 = 1$, and run the SVE algorithm. Result:

$$\sum_{k=0}^{M-1} \alpha_k |\boldsymbol{u}_k\rangle|\widetilde{\sigma}_k\rangle + O(\varepsilon). \tag{4.37}$$

Step 2 Multiply each term of (4.37) separately with scalar $\widetilde{\sigma}_k^{-1}$ for $0 \le k < M$, with register $|\widetilde{\sigma}_k\rangle$ as the control. Result up to normalization:

$$\sum_{k=0}^{M-1} \widetilde{\sigma}_k^{-1} \alpha_k |\boldsymbol{u}_k\rangle|\widetilde{\sigma}_k\rangle + O(\varepsilon). \tag{4.38}$$

Step 3 Undo Step 1 to eliminate register $|\widetilde{\sigma}_k\rangle$. Then the solution state is obtained:

$$\sum_{k=0}^{M-1} \widetilde{\sigma}_k^{-1} \alpha_k |\boldsymbol{u}_k\rangle + O(\varepsilon). \tag{4.39}$$

4.3 Application: Boolean Equation Solving

A variable/solution is said to be Boolean, if it takes value only in $\{0, 1\}$. A Boolean polynomial is an $\mathbb{F}_2$-polynomial in n Boolean variables. The coefficients of a Boolean polynomial are all 1, and any Boolean monomial is linear in each of its variables. Solving Boolean polynomial system is an important task in many problems, such as the subset sum problem, and the graph isomorphism problem.

In 2017, Faugère[46] proposed a quantum algorithm for solving Boolean quadratic equations, by applying Grover's algorithm in some step of the classical algorithm of Bardet, et al.[243]. The complexity of this algorithm is $O(2^{0.462n})$, where n is the number of indeterminates. It is better than direct application of Grover's algorithm, which would have complexity $O(2^{0.5n})$.

In the same year, Chen and Gao[120] proposed a quantum algorithm to solve general Boolean equations over $\mathbb{F}_2$. The complexity of this algorithm is $\widetilde{O}((n^{3.5} + T^{3.5})\kappa^2 \log(1/\varepsilon))$, where T is the total number of terms of the input polynomials, κ is the maximal condition number of the converted complex linear systems, and ε is the precision. The key idea in Chen and Gao's algorithm is to convert a Boolean polynomial system into an equivalent sparse linear system over the complex numbers, then solve it with HHL linear solver.

Below we introduce Chen and Gao's algorithm[120]. Let the input Boolean polynomial system be $\mathcal{F}$. Since it is trivial to decide whether taking value 0 for all variables, or taking value 1 for all variables, is a solution to $\mathcal{F}$, we simply assume that none of $(0, 0, \cdots, 0)$ and $(1, 1, \cdots, 1)$ is a solution to $\mathcal{F}$.

Step 1 Convert the input Boolean polynomial system $\mathcal{F}$ into a projection-equivalent 3-sparse Boolean polynomial system, denoted by $\mathcal{F}_3$.

By introducing new Boolean variables to represent Boolean binomials, any Boolean polynomial can be extended to a Boolean trinomial system, or 3-sparse polynomial system. In this way, $\mathcal{F}$ is extended to a 3-sparse Boolean polynomial system, denoted by $\mathcal{F}_3$. The projection of the $\mathbb{F}_2$-variety defined by $\mathcal{F}_3$ onto the component space of the original Boolean variables, is just the $\mathbb{F}_2$-variety of the input polynomial system $\mathcal{F}$.

For example, if $\mathcal{F} = \{x_1x_2 + x_2 + x_3 + x_4\}$, by introducing new Boolean variable $x_5 := x_1x_2 + x_2$, $\mathcal{F}$ is extended to $\mathcal{F}_3 = \{x_5 + x_1x_2 + x_2, x_5 + x_3 + x_4\}$. The Boolean solutions of $\mathcal{F}_3$ when projected onto the components x_1, x_2, x_3, x_4, give all the Boolean solutions of $\mathcal{F}$.

Step 2 Convert 3-sparse Boolean polynomial system $\mathcal{F}_3$ into an equivalent 6-sparse $\mathbb{C}$-polynomial system, denoted by $\mathcal{F}_{3,\mathbb{C}}$.

A $\mathbb{C}$-valued variable x_i is Boolean if and only if

$$x_i^2 - x_i = 0, \text{ or equivalently, } x_i(x_i - 1) = 0. \tag{4.40}$$

A 3-sparse Boolean polynomial is a polynomial of at most three terms and each term has coefficient 1. Any 3-sparse Boolean polynomial f_j equals zero in $\mathbb{F}_2$ if and only if when f_j is taken as a $\mathbb{C}$-polynomial, then either $f_j = 0$ or $f_j = 2$ when all its variables x_i satisfy (4.40).

By replacing every polynomial $f_j \in \mathcal{F}_3$ with the polynomial of

$$f_j(f_j-2)=0, \tag{4.41}$$

then adding into $\mathcal{F}_3$ the polynomial of (4.40) for every variable x_i of $\mathcal{F}_3$, an enlarged $\mathbb{C}$-polynomial system is obtained, whose elements are 6-sparse, because if $f_j = a+b+c$ is a trinomial, where monomials a, b, c take values in $\{0, 1\}$, then

$$f_j(f_j-2)=a^2+b^2+c^2+2ab+2ac+2bc-2a-2b-2c=2ab+2ac+2bc-a-b-c. \tag{4.42}$$

The enlarged polynomial system is denoted by $\mathcal{F}_{3,\mathbb{C}}$. The $\mathbb{F}_2$-variety of the Boolean polynomial system $\mathcal{F}_3$ is identical with the $\mathbb{C}$-variety of its enlargement $\mathcal{F}_{3,\mathbb{C}}$.

Step 3 Convert 6-sparse $\mathbb{C}$-polynomial system $\mathcal{F}_{3,\mathbb{C}}$ into an equivalent sparse $\mathbb{C}$-linear system, denoted by $\mathcal{F}_{3,\mathbb{C},\mathcal{M}}$.

The original Boolean polynomial system $\mathcal{F}$ has n variables and $O(T)$ polynomials, the converted 3-sparse Boolean polynomial system $\mathcal{F}_3$ has $O(n+T)$ variables and $O(T)$ polynomials. Suppose $\mathcal{F}_{3,\mathbb{C}}$ contains r polynomials in N variables, then $r = O(n+T)$ and $N = O(n+T)$.

Given an integer $D > 0$, all monomials of total degree $\leq D$ in the variables of $\mathcal{F}_{3,\mathbb{C}}$ span a $\mathbb{C}$-linear space of dimension $M := \binom{D+N}{N}$. Observe that a solution to a polynomial equation $f_i = 0$ is always a solution to $f_i g = 0$ for any monomial g. Any polynomial f_i of $\mathcal{F}_{3,\mathbb{C}}$ can generate a set of polynomials of degree $\leq D$ when multiplied with suitable monomials. Denote by $\mathcal{F}_{3,\mathbb{C},\mathcal{M}}$ all the polynomials derived from the polynomials of $\mathcal{F}_{3,\mathbb{C}}$ in this way.

Each polynomial in $\mathcal{F}_{3,\mathbb{C},\mathcal{M}}$, when set to be zero, is of the form $\sum_{i=1}^{M-1} \lambda_i m_i = -\lambda_0$, where the m_i are monomials of degree between 1 and D, and λ_0 is the constant term of the polynomial. Then polynomial system $\mathcal{F}_{3,\mathbb{C},\mathcal{M}}$ when taken as equations, can be written as a linear system of the form

$$A(m_1, \cdots, m_{M-1})^{\mathrm{T}} = \boldsymbol{b}, \tag{4.43}$$

where $\boldsymbol{b}$ is a vector of dimension $\sum_{i=1}^{r} \binom{D+n-\deg f_i}{n}$, and the summation runs over all $f_i \in \mathcal{F}_{3,\mathbb{C}}$.

When D takes the complete solving degree[120] of $\mathcal{F}_{3,\mathbb{C}}$, then (4.43) is the *Macaulay linear system* associated with $\mathcal{F}_{3,\mathbb{C}}$, and solving the linear system gives all the $\mathbb{C}$-solutions to $\mathcal{F}_{3,\mathbb{C}}$. By a delicate corollary of Lazard's Lemma[120], for polynomial system $\mathcal{F}_{3,\mathbb{C}}$, its complete solving degree is $O(n+T)$. Furthermore, the modified Macaulay matrix A has both row sparsity and column sparsity $O(n+T)$.

Step 4 Solve linear system (4.43) by HHL linear solver within error bound $\sqrt{\varepsilon/(n+T)}$.

The $\mathbb{C}$-variety of $\mathcal{F}_{3,\mathbb{C}}$ is obviously 0-dimensional. Let all the zeros of $\mathcal{F}_{3,\mathbb{C}}$ be $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_w$. Then each $\boldsymbol{z}_i$ is a $(0, 1)$-string of length $M-1$. Since HHL algorithm returns the least-square solution, a result of [120] shows that with high probability, the algorithm returns an approximate of a state that can be written as

$$\sum_{i=1}^{w} \eta_i |m_1(\boldsymbol{z}_i), \cdots, m_{M-1}(\boldsymbol{z}_i)\rangle \tag{4.44}$$

up to normalization, where the $\eta_i \in \mathbb{C}$ satisfy $\sum_{i=1}^{w} \eta_i = 1$.

Step 5 Make measurement to the solution state to obtain a solution of value 1 for at least one variable of $\mathcal{F}_{3,\mathbb{C}}$.

For example, if ground state $|1, 0, \cdots, 0\rangle$ is a measured result of (4.44), then $m_1(\boldsymbol{z}_i) = 1$ for some zero point $\boldsymbol{z}_i$ of $\mathcal{F}_{3,\mathbb{C}}$. Since m_1 is a monomial in $(0,1)$-valued variables, all its variables must take value 1 at point $\boldsymbol{z}_i$. The variables in m_1 are thus solved.

Step 6 Update polynomial system $\mathcal{F}_{3,\mathbb{C}}$ by substituting value 1 into solved variables.

Repeat the above procedure from Step 3 to Step 6, until no solution with value 1 is found for $\mathcal{F}_{3,\mathbb{C}}$. Then a complete solution is found for the original system $\mathcal{F}$, where the variables remaining unsolved take value 0.

Extending Chen and Gao's algorithm to solve polynomial equations over a finite field is straightforward[121]. Let the finite field be $\mathbb{F}_m$. In Step 2 of the above algorithm, for any variable X_i taking value in $\mathbb{F}_m$, (4.40) is replaced by

$$X_i(X_i - 1) \cdots (X_i - m + 1) = 0, \tag{4.45}$$

and (4.41) for 3-sparse polynomial f_j over $\mathbb{F}_m$, is replaced by

$$f_j(f_j - m)(f_j - 2m) \cdots (f_j - qm) = 0, \text{ where } q < 3m^{\deg f_j}. \tag{4.46}$$

Furthermore, by introducing new variables, any polynomial equation can be replaced by a set of quadratic ones. By the binary expression of integers, a variable taking value in $\mathbb{F}_m$ can be replaced by a set of Boolean variables.

In [121], Chen, et al. also considered the following optimization problem over a finite field: for variables $X = \{x_1, \cdots, x_n\}$ and $Y = \{y_1, \cdots, y_t\}$, given functions $f_1, \cdots, f_r \in \mathbb{F}_m[X]$ and $h, g_1, \cdots, g_s \in \mathbb{Z}[X, Y]$, given positive integers $b_1, \cdots, b_s$ and $u_1, \cdots, u_t$, solve for

$$\text{Minimize}_{X,Y}\ h(X,Y), \quad \text{subject to} \begin{cases} f_i(X) = 0 \mod m, & i = 1, 2, \cdots, r; \\ 0 \le g_j(X,Y) \le b_j, & j = 1, 2, \cdots, s; \\ 0 \le y_k \le u_k, & k = 1, 2, \cdots, t. \end{cases} \tag{4.47}$$

Since set X and the range of values of the y_k are both finite, so is the range of values of function h. Finding $\min_{X,Y} h(X,Y)$ can be done by solving $h(X,Y) = h_i$, where h_i ranges over all possible values of function h, starting from the smallest one. Each inequality $0 \le g_j \le b_j$ can be replaced by equality $g_j(g_j - 1) \cdots (g_j - b_j) = 0$. By introducing binary expressions for bounded integer variables, the optimization problem over a finite field is reduced to $\mathbb{C}$-polynomial equation solving for Boolean variables.

4.4 Application in Machine Learning: Supervised Classification

Let there be two subsets (classes, or clusters) of vectors $V = \{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_M\}$ and $W = \{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_M\}$ in $\mathbb{R}^N$. Problem: Given $\boldsymbol{u} \in \mathbb{R}^N$, determine whether $\boldsymbol{u}$ should be assigned to V

or W, based on the distances from $\boldsymbol{u}$ to the means of V and W respectively:

$$\text{dist}(\boldsymbol{u}, V) := \left\| \boldsymbol{u} - \frac{1}{M}\sum_{j=1}^{M} \boldsymbol{v}_j \right\|; \qquad \text{dist}(\boldsymbol{u}, W) := \left\| \boldsymbol{u} - \frac{1}{M}\sum_{j=1}^{M} \boldsymbol{w}_j \right\|. \tag{4.48}$$

This is the *nearest-mean classifier*, the simplest among multivariate parameter classifiers[232]. Its complexity rests in the evaluation of (4.48). Classical methods to compute the distances cost $O(MN)$.

In 2013, a quantum algorithm for the problem was proposed by Lloyd, et al.[127], which has complexity $O((\log MN)/\varepsilon)$, where ε is the precision. It achieves exponential speedup over classical algorithms. Below we introduce this algorithm.

Consider evaluating $\text{dist}^2(\boldsymbol{u}, V)$ by (4.48). In quantum computing, evaluating an inner product can be done efficiently by swap test, as long as the classical vector $\boldsymbol{u} - M^{-1}\sum_{j=1}^{M} \boldsymbol{v}_j$ can be converted into its quantum state form, or more accurately, a quantum state of the form

$$|\psi\rangle = t\left(\|\boldsymbol{u}\|\,|\boldsymbol{u}\rangle - \frac{1}{M}\sum_{j=1}^{M} \|\boldsymbol{v}_j\|\,|\boldsymbol{v}_j\rangle \right)|0\rangle + (\cdots)|0\rangle^{\perp}, \tag{4.49}$$

where $t > 0$ is a known scalar.

Step 1 Prepare the initial state

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}|0\rangle\big(|0\rangle|0\rangle - |1\rangle|\varpi_M\rangle\big) = \frac{1}{\sqrt{2}}\left[|0\rangle|0\rangle|0\rangle - \frac{1}{\sqrt{M}}\sum_{j=1}^{M}|0\rangle|1\rangle|j\rangle \right]. \tag{4.50}$$

Step 2 Use the last register to control in the first register of $|\psi_1\rangle$ the preparation of $\|\boldsymbol{u}\|\,|\boldsymbol{u}\rangle$ when the register is in state $|0\rangle$, and $\|\boldsymbol{v}_j\|\,|\boldsymbol{v}_j\rangle$ when the register is in state $|j\rangle$ for $1 \le j \le M$.

Let $\boldsymbol{u} = (u_0, \cdots, u_{N-1})$ and $\boldsymbol{v}_j = (v_{j0}, \cdots, v_{j(N-1)})$ for $1 \le j \le M$. Denote

$$t = \frac{1}{\max_{j,k}(|u_k|, |v_{jk}|)}. \tag{4.51}$$

Assume that t is given. t can be replaced by a known lower bound of the $1/|u_k|$ and $1/|v_{jk}|$. In (3.32) of the proving procedure of Proposition 3.4, quantum state $\frac{t\|\boldsymbol{u}\|}{\sqrt{N}}|\boldsymbol{u}\rangle|0\rangle + (\cdots)|0\rangle^{\perp}$ is prepared efficiently. Similarly, $\frac{t\|\boldsymbol{v}_j\|}{\sqrt{N}}|\boldsymbol{v}_j\rangle|0\rangle + (\cdots)|0\rangle^{\perp}$ can be prepared simultaneously.

The result of Step 2 is

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}\left[\left(\frac{t\|\boldsymbol{u}\|}{\sqrt{N}}|\boldsymbol{u}\rangle|0\rangle + (\cdots)|0\rangle^{\perp} \right)|0\rangle|0\rangle - \frac{1}{\sqrt{M}}\sum_{j=1}^{M}\left(\frac{t\|\boldsymbol{v}_j\|}{\sqrt{N}}|\boldsymbol{v}_j\rangle|0\rangle + (\cdots)|0\rangle^{\perp} \right)|1\rangle|j\rangle \right]. \tag{4.52}$$

The next to the last register of $|\psi_2\rangle$ has one qubit, and takes state value $|0\rangle$ in the first part of the expression, and state value $|1\rangle$ in the second part.

Step 3 To change $|j\rangle$ in the last register of $|\psi_2\rangle$ into $|0\rangle$ for all $1 \le j \le M$, use the next to the last register of $|\psi_2\rangle$ as the control register, so that if the control register is in state $|0\rangle$, then do nothing, else apply Hadamard gate to the last register.

Result:

$$|\psi_3\rangle = \frac{t}{\sqrt{2N}}\Bigg[\|\boldsymbol{u}\|\,|\boldsymbol{u}\rangle|0\rangle|0\rangle|0\rangle - \frac{1}{M}\sum_{j=1}^{M}\|\boldsymbol{v}_j\|\,|\boldsymbol{v}_j\rangle|0\rangle|1\rangle|0\rangle + (\cdots)|0\rangle^{\perp}|0\rangle|0\rangle$$
$$+(\cdots)|0\rangle|1\rangle|0\rangle^{\perp} + (\cdots)|0\rangle^{\perp}|1\rangle|0\rangle + (\cdots)|0\rangle^{\perp}|1\rangle|0\rangle^{\perp}\Bigg]. \tag{4.53}$$

Step 4 Apply Hadamard operator to the next to the last register of $|\psi_3\rangle$. Result:

$$|\psi_4\rangle = \frac{t}{2\sqrt{N}}\left[\|\boldsymbol{u}\|\,|\boldsymbol{u}\rangle - \frac{1}{M}\sum_{j=1}^{M}\|\boldsymbol{v}_j\|\,|\boldsymbol{v}_j\rangle\right]|0,0,0\rangle + (\cdots)|0,0,0\rangle^{\perp}. \tag{4.54}$$

Then quantum state $\|\boldsymbol{u}\|\,|\boldsymbol{u}\rangle - \frac{1}{M}\sum_{j=1}^{M}\|\boldsymbol{v}_j\|\,|\boldsymbol{v}_j\rangle$ is prepared.

Step 5 In (4.54), the probability of observing state $|0,0,0\rangle$ in the last three registers is

$$\frac{t^2}{4N}\left(\boldsymbol{u} - \frac{1}{M}\sum_{j=1}^{M}\boldsymbol{v}_j\right)^2 = \frac{t^2}{4N}\mathrm{dist}^2(\boldsymbol{u}, V). \tag{4.55}$$

Use swap test (Lemma 3.2) to estimate $t^2\mathrm{dist}^2(\boldsymbol{u}, V)/(4N)$, from which $\mathrm{dist}^2(\boldsymbol{u}, V)$ is obtained.

4.5 Application in Machine Learning: Linear Regression

Given a set of samples $\{(\boldsymbol{x}_i, y_i) \mid 1 \le i \le N\}$, where each $\boldsymbol{x}_i \in \mathbb{C}^M$ is a column vector, and each $y_i \in \mathbb{C}$ is a data, linear regression aims at finding a vector $\boldsymbol{c} \in \mathbb{C}^M$ minimizing $\sum_{i=1}^{N}|\boldsymbol{x}_i^{\dagger}\boldsymbol{c} - y_i|^2$. By differentiation, the problem is reduced to solving the following linear system:

$$X^{\dagger}X\boldsymbol{c} = X^{\dagger}\boldsymbol{y}, \tag{4.56}$$

where $X^{\dagger} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N)$, and $\boldsymbol{y} = (y_1, \cdots, y_N)^{\mathrm{T}}$.

Linear regression plays an important role in (supervised) machine learning[244, 245]. The purpose of linear regression is to find a linear function that reflects the global trend of the data evolution, based on which a prediction can be made. Since HHL algorithm returns the least-square solution of a linear system, it can be applied directly to linear regression[124]. Besides HHL algorithm, other quantum algorithms for linear regression include [109, 125, 126, 228].

Linear regression is also an example in which the quantum state solution of a quantum linear solver is sufficient to solve the problem. Suppose a solution state $|\boldsymbol{c}\rangle$ is obtained from a quantum linear solver. For a new vector $\boldsymbol{d} \in \mathbb{C}^M$, the prediction on the new data is done by evaluating the inner product $\langle\boldsymbol{c}|\boldsymbol{d}\rangle$, which can be estimated by swap test.

Regularization of ill-conditioned linear regression.

In solving linear system (4.56), if $X^{\dagger}X$ is not invertible, then the linear solver may fail if $X^{\dagger}\boldsymbol{y}$ is not in the well-conditioned region of $X^{\dagger}X$, as mentioned in Remark 4.1.

One method to overcome this difficulty is regularization[245]. A new term λI is added to $X^{\dagger}X$, where λ is the *regularization parameter*, and I is the identity matrix. The linear system is changed into $(X^\dagger X + \lambda I)\widetilde{\boldsymbol{c}} = X^\dagger \boldsymbol{y}$ for new vector variable $\widetilde{\boldsymbol{c}}$ that is taken as an approximate of $\boldsymbol{c}$. The choice of parameter λ is technical.

Locally weighted linear regression.

In data fitting, a more prevalent method is locally weighted linear regression (or "piecewise" linear regression)[246, 247]. In this method, linear regression is done locally based on the samples close to the new vector whose data is to be predicted.

Locally weighted linear regression is equivalent to solving the following linear system:

$$X^\dagger W X \boldsymbol{c} = X^\dagger W \boldsymbol{y}, \tag{4.57}$$

where diagonal matrix $W = \mathrm{diag}(w_0, \cdots, w_{N-1})$ serves as a filter, in which $w_i \geq 0$ for $0 \leq i < N$. A possible choice of w_i is the Gauss kernel function[244, 248]. Usually W is a low-rank matrix.

Set $X_{\mathrm{new}} = \sqrt{W}X$, and $\boldsymbol{y}_{\mathrm{new}} = \sqrt{W}\boldsymbol{y}$. Then (4.57) can be written as

$$X_{\mathrm{new}}^\dagger X_{\mathrm{new}} \boldsymbol{c} = X_{\mathrm{new}}^\dagger \boldsymbol{y}_{\mathrm{new}}. \tag{4.58}$$

Since W is low-rank, so is X_{new}. According to [112, 127], low-rank Hermitian matrices can be simulated efficiently. So HHL algorithm is efficient for locally weighted linear regression.

4.6 Application in Machine Learning: Least-Square Support Vector Machine (SVM)

Given a set of training examples $\{\boldsymbol{x}_i \in \mathbb{R}^N,\ 1 \leq i \leq M\}$, each with a label $y_i \in \{1, -1\}$ for $1 \leq i \leq M$ to mark the example as belonging to one of two classes, the SVM problem is to find a hyperplane that divides the points of separate classes with maximal margin[232, 245].

Denote the hyperplane that divides the two classes by $\boldsymbol{w}\cdot\boldsymbol{x} - b = 0$, where $b \in \mathbb{R}$ and $\boldsymbol{w} \in \mathbb{R}^N$. Formally, the hyperplane is constructed so that $\boldsymbol{w}\cdot\boldsymbol{x}_j - b \geq 1$ if $y_j = 1$, and $\boldsymbol{w}\cdot\boldsymbol{x}_j - b \leq -1$ if $y_j = -1$. The points on the hyperplanes $\boldsymbol{w}\cdot\boldsymbol{x} - b = \pm 1$ are called support points. The SVM problem aims at finding a hyperplane $\boldsymbol{w}\cdot\boldsymbol{x} - b = 0$ that maximizes the distance between the two parallel hyperplanes $\boldsymbol{w}\cdot\boldsymbol{x} - b = \pm 1$.

The distance from any support point to hyperplane $\boldsymbol{w}\cdot\boldsymbol{x} - b = 0$ is $1/\|\boldsymbol{w}\|$. So the SVM problem can be stated as the following quadratic convex optimization:

$$\min_{b\in\mathbb{R}, \boldsymbol{w}\in\mathbb{R}^N} \ \|\boldsymbol{w}\|^2, \quad \text{s.t.} \ \ y_j(\boldsymbol{w}\cdot\boldsymbol{x}_j - b) \geq 1, \ \ j = 1, 2, \cdots, M. \tag{4.59}$$

When all the inequalities in (4.59) are replaced by equalities, the optimization problem is called *least-square SVM*[249]:

$$\min_{b, e_j\in\mathbb{R}, \boldsymbol{w}\in\mathbb{R}^N} \ \left(\|\boldsymbol{w}\|^2 + \gamma \sum_{j=1}^{M} e_j^2\right), \quad \text{s.t.} \ \ y_j(\boldsymbol{w}\cdot\boldsymbol{x}_j - b) = 1 - e_j, \ \ j = 1, 2, \cdots, M, \tag{4.60}$$

where $\gamma > 0$ is determined by the user, and the e_j are slack variables that are indispensable if the separating hyperplane does not exist: They measure the deviation of a data point from the ideal condition of pattern separability.

The quadratic programming (4.60) can be reduced to a linear system by introducing Lagrange multipliers $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_M)^{\mathrm{T}}$. The Lagrange function is

$$L = \frac{\|\boldsymbol{w}\|^2}{2} + \frac{\gamma}{2}\sum_{j=1}^{M} e_j^2 - \sum_{j=1}^{M} \lambda_j \Big[(\boldsymbol{w}\cdot\boldsymbol{x}_j - b)y_j - 1 + e_j\Big]. \tag{4.61}$$

In the optimization, the partial derivatives of L with respect to its arguments are all zero:

$$\begin{cases} \nabla_{\boldsymbol{w}} L = \boldsymbol{w} - \sum\limits_{j=1}^{M} \lambda_j y_j \boldsymbol{x}_j = 0, \\ \partial L/\partial e_j = \gamma e_j - \lambda_j = 0, \\ \partial L/\partial b = \sum\limits_{j=1}^{M} \lambda_j y_j = 0, \\ \partial L/\partial \lambda_j = (\boldsymbol{w}\cdot\boldsymbol{x}_j - b)y_j - 1 + e_j = 0. \end{cases} \tag{4.62}$$

The first equation of (4.62) gives $\boldsymbol{w} = \sum_{j=1}^{M} \lambda_j y_j \boldsymbol{x}_j$. Given a new vector $\boldsymbol{z} \in \mathbb{R}^M$, its classification by hyperplane $\boldsymbol{w}\cdot\boldsymbol{x} - b = 0$ is determined by the sign of

$$\boldsymbol{w}\cdot\boldsymbol{z} - b = -b + \sum_{j=1}^{M} \lambda_j y_j \boldsymbol{x}_j \cdot \boldsymbol{z}. \tag{4.63}$$

Estimating the sign of the expression on the right side can be done in the same way as evaluating $\mathrm{dist}^2(\boldsymbol{u}, V)$ by (4.48) in supervised classification. We sketch some details below.

Eliminating $\boldsymbol{w}$ and the e_j from (4.62) leads to the following linear system:

$$\begin{bmatrix} 0 & \boldsymbol{y}^{\mathrm{T}} \\ \boldsymbol{y} & K + \gamma^{-1} I_M \end{bmatrix} \begin{bmatrix} -b \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}, \tag{4.64}$$

where

$$K = (y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j)_{i,j=1,2,\cdots,M}, \quad \boldsymbol{y} = (y_1, \cdots, y_M)^{\mathrm{T}}, \quad \mathbf{1} = (1, \cdots, 1)^{\mathrm{T}} \in \mathbb{R}^M. \tag{4.65}$$

Step 1 By HHL algorithm, a quantum state solution to (4.64) can be obtained, and is in the form

$$|-b, \boldsymbol{\lambda}\rangle := \frac{1}{\sqrt{b^2 + \sum_{j=1}^{M} \lambda_j^2}} \Big[-b|0\rangle + \sum_{j=1}^{M} \lambda_j |j\rangle \Big]. \tag{4.66}$$

Step 2 Similar to the construction of (4.52), from (4.66) the following quantum state can be constructed:

$$|\psi_{-b,\lambda}\rangle = \frac{1}{\sqrt{b^2 + \sum_{j=1}^{M} \lambda_j^2 \|\boldsymbol{x}_j\|^2}} \Big[-b|0\rangle|0\rangle + \sum_{j=1}^{M} \lambda_j y_j \|\boldsymbol{x}_j\|\, |j\rangle|\boldsymbol{x}_j\rangle \Big]. \tag{4.67}$$

Similar to the construction of (4.54), the following quantum state ("query state") can be constructed simultaneously with the construction of $|\psi_{-b,\lambda}\rangle$:

$$|\psi_{\boldsymbol{z}}\rangle = \frac{1}{\sqrt{M\|\boldsymbol{z}\|^2+1}}\left[|0\rangle|0\rangle + \sum_{j=1}^{M}\|\boldsymbol{z}\|\,|j\rangle|\boldsymbol{z}\rangle\right]. \tag{4.68}$$

Step 4 By (4.67), (4.68) and (4.63),

$$\langle\psi_{-b,\lambda}|\psi_{\boldsymbol{z}}\rangle = \mu(-b+\boldsymbol{w}\cdot\boldsymbol{z}), \text{ for some } \mu>0. \tag{4.69}$$

By swap test, the inner product of $|\psi_{-b,\lambda}\rangle$ and $|\psi_{\boldsymbol{z}}\rangle$ can be estimated, by whose sign $\boldsymbol{z}$ is classified into the correct class.

Applying HHL algorithm to solve (4.64) needs efficient Hamiltonian simulation of the coefficient matrix. In [129], a method was presented to do this. The method was inspired by the work of [128] on quantum principal component analysis.

5 Quantum Walk

A quantum walk is the quantum counterpart of a random walk[95]. To introduce quantum walk, first recall some basic facts of random walk[250, 251].

5.1 Basic Terminology in Markov Chain

A *Markov process* is a random process that takes a distribution on some state space Ω, and maps it to another distribution on Ω, such that the distribution of the next state depends only on the current state. A *random walk* (or *Markov chain*, or *Markov diffusion*) (Ω, P), is a Markov process with a stochastic transition matrix $P=(p_{uv})$ defined on a discrete state space Ω, such that for any $u,v\in\Omega$, p_{uv} gives the probability of moving to state v from state u.

A Markov chain is *symmetric* if its transition matrix is symmetric. A Markov chain is *state-transitive* if there is a transitive permutation group acting on the state space that leaves the transition matrix invariant. A random walk is called a *uniform diffusion* if all entries of the transition matrix are identical.

The *period* of state u is the minimal nonzero step to return to u when starting from u. A Markov chain is said to be *aperiodic* if the GCD of all the periods of the states is 1. A Markov chain is *irreducible* if every state is reachable from every other state. It is *ergodic* if it is both irreducible and aperiodic.

A *stationary distribution* $\boldsymbol{\pi}^{\mathrm{T}}$ of Markov chain (Ω, P) is a probability distribution on Ω such that when $\boldsymbol{\pi}^{\mathrm{T}}$ is taken a row vector, then $\boldsymbol{\pi}^{\mathrm{T}}P=\boldsymbol{\pi}^{\mathrm{T}}$. As $\boldsymbol{\pi}^{\mathrm{T}}$ is a left eigenvector of matrix P with eigenvalue 1, if the current state is distributed according to $\boldsymbol{\pi}^{\mathrm{T}}$, the next state follows the same distribution.

Any irreducible Markov chain has a unique stationary distribution. For any irreducible and symmetric Markov chain, its stationary distribution is uniform.

For an irreducible Markov chain (Ω, P), denote $|\Omega|=N$, and denote the stationary distribution by row vector

$$\boldsymbol{\pi}^{\mathrm{T}} = (\boldsymbol{\pi}^{\mathrm{T}}(u))_{u\in\Omega} := (\pi_1,\cdots,\pi_N). \tag{5.1}$$

The *time-reversed Markov chain* of (Ω, P) is a Markov chain $(\Omega, P^\star)$, where $P^\star = (p^\star_{uv})_{u,v\in\Omega}$ is defined by

$$p^\star_{uv} := \frac{\pi_v}{\pi_u} p_{vu}. \tag{5.2}$$

(Ω, P) is said to be *reversible* if $P^\star = P$, i.e., for every pair of states $u, v \in \Omega$,

$$\pi_u P_{uv} = \pi_v P_{vu}. \tag{5.3}$$

A random walk on an undirected graph is always reversible.

The *discriminant matrix* of matrix P is

$$D(P) := (\sqrt{p_{uv} p^\star_{vu}})_{u,v\in\Omega}. \tag{5.4}$$

By $\sqrt{p_{uv} p^\star_{vu}} = \sqrt{\pi_u} p_{uv} \sqrt{\pi_v}^{-1}$,

$$D(P) = \sqrt{\mathrm{diag}(\boldsymbol{\pi}^{\mathrm{T}})} P \sqrt{\mathrm{diag}(\boldsymbol{\pi}^{\mathrm{T}})}^{-1}, \tag{5.5}$$

where $\mathrm{diag}(\boldsymbol{\pi}^{\mathrm{T}})$ is the diagonal matrix whose diagonal entries are the components of vector $\boldsymbol{\pi}^{\mathrm{T}}$. Matrix $D(P)$ is symmetric if and only if (Ω, P) is reversible.

The distance between two distributions (row vectors) $\boldsymbol{\rho}^{\mathrm{T}} = (\rho_u)_{u=1,2,\cdots,N}, {\boldsymbol{\rho}'}^{\mathrm{T}} = (\rho'_u)_{u=1,2,\cdots,N}$ on Ω, is measured by their *total variation distance*:

$$\|\boldsymbol{\rho}^{\mathrm{T}} - {\boldsymbol{\rho}'}^{\mathrm{T}}\|_{\mathrm{tv}} := \frac{1}{2} \sum_{u\in\Omega} |\rho_u - \rho'_u|. \tag{5.6}$$

The total variation distance between $\boldsymbol{\rho}^{\mathrm{T}}, {\boldsymbol{\rho}'}^{\mathrm{T}}$ is half the L_1-norm of their difference.

For an ergodic Markov chain (Ω, P), the *mixing time* T_{mix} is the minimal number of steps for any initial distribution to approximate the stationary distribution:

$$T_{\mathrm{mix}} := \min\{t \in \mathbb{N} \,\big|\, \|\boldsymbol{\rho}^{\mathrm{T}} P^t - \boldsymbol{\pi}^{\mathrm{T}}\|_{\mathrm{tv}} \le 1/4, \forall \text{ initial distribution } \boldsymbol{\rho}^{\mathrm{T}}\}. \tag{5.7}$$

In (5.7), it seems more appropriate to measure the approximation of $\boldsymbol{\pi}^{\mathrm{T}}$ by $\boldsymbol{\rho}^{\mathrm{T}} P^t$ with some $\varepsilon \ll 1$, by defining the "ε-mixing time" T_ε to be the minimal natural number t such that for any initial distribution $\boldsymbol{\rho}^{\mathrm{T}}$, $\|\boldsymbol{\rho}^{\mathrm{T}} P^t - \boldsymbol{\pi}^{\mathrm{T}}\|_{\mathrm{tv}} \le \varepsilon$. In fact, $T_{\mathrm{mix}} \le T_\varepsilon \le \lceil \log_2(1/\varepsilon) \rceil T_{\mathrm{mix}}$. So $T_\varepsilon = \widetilde{\Theta}(T_{\mathrm{mix}})$, and the two mixing times can be taken to be equivalent.

An important concept in Markov chain is the spectral gap of the transition matrix P. Let $\lambda_1, \cdots, \lambda_N$ be the multiset of all the eigenvalues of P in norm-decreasing order. It is a classical result[252] that $1 = \lambda_1 \ge |\lambda_2| \ge \cdots \ge |\lambda_N|$. The *spectral gap* of P is defined as

$$\delta := 1 - |\lambda_2| \ge 0. \tag{5.8}$$

If P is ergodic, then $\delta > 0$.

The spectral gap is closely related to the mixing time T_{mix}. If (Ω, P) is reversible and ergodic, with spectral gap δ and stationary distribution $\boldsymbol{\pi}^{\mathrm{T}} = (\pi_u)_{u\in\Omega}$, then

$$\left(\frac{1}{\delta} - 1\right) \log \frac{3}{2} \le T_{\mathrm{mix}} \le \frac{1}{\delta} \log \frac{3}{\min_{u\in\Omega} \pi_u}. \tag{5.9}$$

When $\boldsymbol{\pi}^{\mathrm{T}}$ is the uniform distribution, then $\min_{u\in\Omega}\pi_u = 1/N$, and $T_{\mathrm{mix}} = \widetilde{\Theta}(1/\delta)$.

The *hitting time* $h(u,v)$ from state u to state v is the expected number of steps to hit v when starting from u. If $\mathcal{M}$ is a set of states, the hitting time $h(u,\mathcal{M})$ can be defined similarly. The *average hitting time* from an initial distribution $\boldsymbol{\rho}^{\mathrm{T}} = (\rho_u)_{u\in\Omega}$ to $\mathcal{M}$ is defined by

$$h := h(\boldsymbol{\rho}^{\mathrm{T}},\mathcal{M}) = \sum_{u\in\Omega}\rho_u h(u,\mathcal{M}). \tag{5.10}$$

It is related to the spectral gap as follows[253]:

$$\frac{1}{\varepsilon} \leq h \leq \frac{1}{\varepsilon\delta}, \quad \text{where } \varepsilon = \frac{|\mathcal{M}|}{N}. \tag{5.11}$$

For random walk (Ω,P), the *absorbing walk* of subset $\mathcal{M}\subset\Omega$ is a random walk $(\Omega,P_{\mathcal{M}})$, with

$$P_{\mathcal{M}} = \begin{pmatrix} P' & P'' \\ 0 & I \end{pmatrix}, \tag{5.12}$$

where matrix P' is obtained from P by deleting all rows and columns indexed by $\mathcal{M}$, and P'' is obtained from P by deleting all rows indexed by $\mathcal{M}$ and columns indexed by $\Omega\setminus\mathcal{M}$. The walker $P_{\mathcal{M}}$ behaves like P but is absorbed by the first state of $\mathcal{M}$ it hits. Absorbing walk is an important tool in designing search algorithms, with $\mathcal{M}$ as the goal.

5.2 Discrete-Time Quantum Walk on the Line

Discrete-time quantum walk on the line is the most extensively studied model of quantum walk. The simplest classical discrete-time random walk on the line consists of a walker and a line. The walker starts in location 0, and moves left with probability 1/2 and right with probability 1/2. The straightforward counterpart would be a quantum process with basis states $|n\rangle$ for all $n\in\mathbb{Z}$, and at each step, a unitary transformation is performed so that the move would be to the left with amplitude a, staying in the same place with amplitude b, and moving to the right with amplitude c, where $|a|^2+|b|^2+|c|^2=1$. However, this is impossible. It is proved in [86] that the only possible transformations are that at each step, the move is either always to the left, or staying in the same place, or always to the right. The same thing happens for quantum walks on general graphs.

To define discrete-time quantum walk on the line correctly, besides the position space $\{|n\rangle,\ n\in\mathbb{Z}\}$ and the (unitary) evolution operators, an additional 1-qubit register called the *coin state* must be introduced, so that the state space includes the coin besides the position. The following is the basis states:

$$\{|n,0\rangle,|n,1\rangle \,\big|\, n\in\mathbb{Z}\}. \tag{5.13}$$

For the evolution operators, at each step there are two unitary operations: A coin flip transformation C deciding in which direction to move, and a shift operator S to make the move, so that one step of quantum walk is the composition SC, for which there are $a,b,c,d\in\mathbb{C}$ satisfying

$|a|^2 + |b|^2 = |c|^2 + |d|^2 = 1$, such that

$$\begin{aligned} C|n,0\rangle &= a|n,0\rangle + b|n,1\rangle, \quad & S|n,0\rangle &= |n-1,0\rangle, \\ C|n,1\rangle &= c|n,0\rangle - d|n,1\rangle, \quad & S|n,1\rangle &= |n+1,1\rangle. \end{aligned} \tag{5.14}$$

The most common choice for C is the Hadamard transform, where $a = b = c = d = 1/\sqrt{2}$. Consider the first three steps of such a quantum walk[87, 90] starting from state $|0,0\rangle$:

$$\begin{aligned} |0,0\rangle &\xrightarrow{C} \frac{1}{\sqrt{2}}(|0,0\rangle + |0,1\rangle) \xrightarrow{S} \frac{1}{\sqrt{2}}(|-1,0\rangle + |1,1\rangle) \\ &\xrightarrow{C} \frac{1}{2}(|-1,0\rangle + |-1,1\rangle + |1,0\rangle - |1,1\rangle) \xrightarrow{S} \frac{1}{2}(|-2,0\rangle + |0,1\rangle + |0,0\rangle - |2,1\rangle) \\ &\xrightarrow{C} \frac{1}{2\sqrt{2}}(|-2,0\rangle + |-2,1\rangle + 2|0,0\rangle - |2,0\rangle + |2,1\rangle) \\ &\qquad\qquad \xrightarrow{S} \frac{1}{2\sqrt{2}}(|-3,0\rangle + |-1,1\rangle + 2|-1,0\rangle - |1,0\rangle + |3,1\rangle). \end{aligned} \tag{5.15}$$

If after the second step the state is measured, then $n = -2$ and $n = 2$ would be found with probability $1/4$ for each, and $n = 0$ be found with probability $1/2$. This is exactly the case in a classical random walk. From the third step on, quantum walk behaves differently from random walk. In random walk, the left and right direction each have probability $1/2$. In the quantum case, due to quantum interference, the probability distribution is non-symmetric and biased towards the left. The difference becomes more striking with the increase of the number of steps.

There are two prominent distinctions of quantum walk from random walk, which exist for almost all choices of coin transformation and initial state[87, 90]:

- In the evolution of the probability distribution of the states, after t steps, the maximum probability is reached for $n \approx \pm t/\sqrt{2}$. In contrast, in classical random walk, the maximum probability is reached for $n \approx 0$.
- There are $\Omega(t)$ locations of n for which the probability of measuring $|n,0\rangle$ or $|n,1\rangle$ is $\Omega(1/t)$. This means that the expected distance between the starting location 0 and the location to make measurement after t steps is $\Omega(t)$. So quantum walk spreads quadratically faster than its classical counterpart. This is key to fast quantum algorithm design.

5.3 Discrete-Time Quantum Walk on Undirected Graph

The following is a simple way to define discrete-time quantum walk on an undirected graph[254–256] with vertices V and edges E: The basis states are

$$\{|v,e\rangle \mid v \in V, e \in E \text{ such that } v \text{ is incident to } e\}. \tag{5.16}$$

The evolution is the composition of a shift operator S and a coin flip transformation C. For the shift operator, if edge e has endpoints v, v', then

$$S|v,e\rangle = |v',e\rangle, \qquad S|v',e\rangle = |v,e\rangle. \tag{5.17}$$

For each vertex v, the coin flip transformation C is performed to all the states $|v, e\rangle$ where e is incident to v. A natural choice of C is Grover's diffusion.

In designing fast quantum algorithms with quantum walk, Johnson graph is an important tool. For any two positive integers n and r with $r < n$, the Johnson graph $J(n, r)$ is the graph whose vertices are all subsets of $\mathcal{N} = \{1, 2, \cdots, n\}$ of cardinality r, and whose edges are such that an edge connects two vertices if and only if the vertices as two subsets of $\mathcal{N}$ differ by exactly one element.

The random walk on $J(n, r)$ can be intuitively described as follows: at every step, if S is the current vertex, then when moving to the next vertex S', since the two vertices must be the two endpoints of an edge, one element of S (the unique element in $S \setminus S'$) is replaced by the only element in $S' \setminus S$. For any $2 < r < n$, the stationary distribution of the random walk on $J(n, r)$ is unique and is the uniform distribution, because $J(n, r)$ is a degree-regular graph. Furthermore, the spectral gap of the transition matrix of the random walk on $J(n, r)$ is $\geq 1/r$.

For a special class of graphs called *glued balanced binary trees*, quantum walk gains exponential speedup over all classical algorithms in hitting time. The simplest graph of this kind is a graph obtained by identifying the leaves of a balanced binary tree with the corresponding leaves of the mirror image of the tree. Suppose that the original tree T lies in the plane, whose leaves are aligned along a vertical line L in the plane. Then the mirror image T' of tree T with respect to line L is another tree whose leaves are identified with those of T. Let the depth of each tree be d, and let the walker go from the root of one tree to the root of the other. For a classical random walker, it takes $\Omega(2^d)$ steps (hitting time) to reach one root from the other. In contrast, Childs, et al.[94] showed that a continuous-time quantum walker can do so in $O(d^2)$ steps.

A more complicated graph is two balanced binary trees of depth d joined by a random cycle of length 2^{d+1} that alternates between the leaves of the two trees. Classical random walkers take $O(2^d)$ steps to find one root from the other, but the quantum walkers in [93, 94] take $O(d^2)$ steps to do so. Furthermore, in [93] it was shown that any classical algorithm (not just classical random walk) requires an exponentially large number of steps.

5.4 Application: Search on Boolean Hypercube

In a Boolean hypercube, there are $N = 2^n$ vertices $\{v_x, x = 0, 1, \cdots, N-1\}$ indexed by n-bit strings (binary expressions of the x). Two vertices v_x and v_y are connected by an edge if and only if the corresponding string forms of x and y differ in exactly one bit. Suppose there is only one marked vertex on the hypercube. Find it.

In quantum walk, the state space is spanned by the basis states

$$\{|x\rangle|i\rangle, x = 0, 1, \cdots, N-1; i = 0, 1, \cdots, n-1\}, \tag{5.18}$$

where $x \in \{0, 1\}^n = \{0, 1, \cdots, N-1\}$ is used to index the vertices of the hypercube, and $i \in \{1, 2, \cdots, n\}$ is used to index the edges by bits. The edge register serves as the coin, and the vertex register serves as the walking space. The value of an n-bit string x at bit i is denoted by x_i. For $x = (x_0, \cdots, x_{n-1})$, where $x_i \in \{0, 1\}$, $\mathrm{flip}(x, i) := (x_0, \cdots, x_{i-1}, 1-x_i, x_{i+1}, \cdots, x_{n-1})$.

The distance between two vertices is the minimal number of edges connecting them. It agrees with the Hamming distance between the two vertices taken as n-bit strings. For an n-bit string x, its *Hamming weight* $|x|$ is the number of nonzero bits, or the distance from 0-string.

Grover search can be used directly on the hypercube. However, due to the spatial structure, a vertex in the hypercube is connected to only $n = \log N$ vertices. At each vertex, only the local Grover's diffusion R_{ϖ_n} is available, and applying the diffusion for once only spreads the superposition of vertex states to the neighbors of the vertex. In other words, the walker following Grover's diffusion *goes only one step* on the hypercube. To reach the farmost vertex, whose distance from the current vertex is n, Grover's diffusion R_{ϖ_n} must be repeated n times, so that all the N states in the superposition $|\varpi_N\rangle$ are accessed. Hence, a factor n is added to the number of iterations $O(\sqrt{N})$ in Grover's search algorithm, giving the *step complexity* (minimal number of steps a most unlucky walker needs to go to make the finding): $O(\sqrt{N}\log N)$.

In 2003, Shenvi, et al.[92] presented an algorithm to make the search by quantum walk in $O(\sqrt{N})$ steps. The following is their algorithm.

Algorithm 5 Quantum search on Boolean hypercube[92]

```
1: [Setup] Generate the superposition state
       $$|\varpi_{N\times n}\rangle := \frac{1}{\sqrt{Nn}}\sum_{x,i}|x\rangle|i\rangle. \qquad (5.19)$$
2: Do the following steps $O(\sqrt{N})$ times:
   (2.1) [Checking, and tossing coin] If vertex $x$ is not marked, then apply Grover's diffusion
         $R_{\varpi_n}$ to register $|i\rangle$, else apply $-I$ to register $|i\rangle$.
   (2.2) [Walking, update of position] Apply shift $S : |x\rangle|i\rangle \mapsto |\mathrm{flip}(x,i)\rangle|i\rangle$.
3: Measure the final state at the first register.
```

The algorithm itself is very simple, and is based on a quantum walk on the hypercube that is taken as an undirected graph. However, the analysis of the correctness of the algorithm is rather involved. Suppose there is a unique marked vertex $m \in \{0, 1, \cdots, N-1\}$ on the hypercube. The evolution operator U is the composition of the shift operator S and the coin transformation C. With respect to the basis states (5.18),

$$C = 2(I_N - |m\rangle\langle m|) \otimes |\varpi_n\rangle\langle\varpi_n| - I_N \otimes I_n. \tag{5.20}$$

A key idea in the analysis is to reduce the quantum walk on the hypercube to a quantum walk on the line. For a line segment $\{0, 1, \cdots, n\}$ of $n+1$ nodes, add a coin with two states $\{R, L\}$ to form the following $2n$ orthonormal basis states: For any $0 \le y < n$ and $0 < z \le n$,

$$|y, R\rangle \coloneqq \frac{1}{\sqrt{(n-y)\binom{n}{y}}}\sum_{|x|=y}\ \sum_{0\le i<n,\, x_i=0}|x, i\rangle, \quad |z, L\rangle \coloneqq \frac{1}{\sqrt{z\binom{n}{z}}}\sum_{|x|=z}\ \sum_{0\le i<n,\, x_i=1}|x, i\rangle. \tag{5.21}$$

Denote by $\varOmega$ the $(2n)$-space they span. Then

$$|\varpi_{N\times n}\rangle = \frac{1}{\sqrt{N}}|0,R\rangle + \frac{1}{\sqrt{N}}|n,L\rangle + \sum_{x=1}^{n-1}\left(\sqrt{\frac{\binom{n-1}{x-1}}{N}}|x,L\rangle + \sqrt{\frac{\binom{n-1}{x}}{N}}|x,R\rangle\right). \tag{5.22}$$

Space $\varOmega$ is invariant under Grover's diffusion R_{ϖ_n} when the latter acts on the edge space of the hypercube. With respect to the basis states (5.21) supplemented with two "forbidden basis states" $|n,R\rangle$ and $|0,L\rangle$, R_{ϖ_n} acts as

$$\sum_{y=0}^{n}|y\rangle\langle y|\otimes\begin{pmatrix}\cos\omega_y & \sin\omega_y\\ \sin\omega_y & -\cos\omega_y\end{pmatrix}, \tag{5.23}$$

where

$$\cos\omega_y = 1-\frac{2y}{n}, \qquad \sin\omega_y = \frac{2}{n}\sqrt{y(n-y)}. \tag{5.24}$$

So the new coin space is invariant under R_{ϖ_n}.

Furthermore, space $\varOmega$ is also invariant under the shift operator S and the evolution $U=SC$. In $\varOmega$, the shift S acts as

$$\sum_{y=0}^{n-1}\left(\big|y,R\rangle\langle y+1,L\big| + \big|y+1,L\rangle\langle y,R\big|\right). \tag{5.25}$$

By (5.20), the restriction of U to $\varOmega$ depends on the specific value of $|m\rangle$. Let the marked state be $|m\rangle=|0\rangle$ for simplicity. In space $\varOmega$, the evolution U acts as

$$\begin{aligned}U':=&-2|1,L\rangle\langle 0,R| + \sum_{x=0}^{n-1}|x,R\rangle\left(-\cos\omega_{x+1}\langle x+1,L| + \sin\omega_{x+1}\langle x+1,R|\right)\\ &+\sum_{x=1}^{n}|x,L\rangle\left(\sin\omega_{x-1}\langle x-1,L| + \cos\omega_{x-1}\langle x-1,R|\right).\end{aligned} \tag{5.26}$$

So the quantum walk on the hypercube induces a quantum walk on the line segment, with basis position states $\{|0\rangle,\cdots,|n\rangle\}$, basis coin states $\{|R\rangle,|L\rangle\}$, and evolution operator U' when $|m\rangle=|0\rangle$. To show the correctness of the algorithm, check how the evolution U accumulates the amplitudes towards the subspace of $\mathbb{C}^N\otimes\mathbb{C}^n$ spanned by $\{|0\rangle|i\rangle,\ i=0,1,\cdots,n-1\}$, when walking on line segment $\varOmega$.

For the initial state $|\varpi_{N\times n}\rangle$,

$$U'|\varpi_{N\times n}\rangle = |\varpi_{N\times n}\rangle - \frac{2}{\sqrt{N}}|1,L\rangle. \tag{5.27}$$

So $\langle\varpi_{N\times n}|U'|\varpi_{N\times n}\rangle = 1-2^{1-n}$, and $|\varpi_{N\times n}\rangle$ is almost an eigenvector of U' with eigenvalue 1.

Define

$$|\phi\rangle := \frac{1}{c}\sum_{x=0}^{n/2-1}\frac{1}{\sqrt{2\binom{n-1}{x}}}\big(|x,R\rangle - |x+1,L\rangle\big), \quad \text{where } c=\sqrt{\sum_{x=0}^{n/2-1}1\Big/\binom{n-1}{x}}. \tag{5.28}$$

It is a unit vector orthogonal to $|\varpi_{N\times n}\rangle$. If this state is available, then by measuring the first register of $|\phi\rangle$ as in Step 3 of Algorithm 5, the probability of getting the marked vertex $|0\rangle$ is

$$\frac{1}{c}\sum_{i=0}^{n-1}\big|\langle 0,i|\phi\rangle\big|^2=\frac{1}{c}\big|\langle 0,R|\phi\rangle\big|^2\geq\frac{1}{2}-O\Big(\frac{1}{n}\Big), \tag{5.29}$$

where $1<c^2<1+2/n$ for sufficiently large n is used. The goal is thus to show that by starting from $|\varpi_{N\times n}\rangle$ and making a number of iterations of U', state $|\phi\rangle$ can be prepared approximately.

It is easy to verify that

$$U'|\phi\rangle=|\phi\rangle-\frac{1}{c\sqrt{2\binom{n-1}{n/2}}}(|n/2-1,R\rangle+|n/2+1,L\rangle). \tag{5.30}$$

So

$$\langle\phi|U'|\phi\rangle=1-\frac{1}{2c^2\binom{n-1}{n/2}}>1-\frac{1}{2(1+\frac{2}{n})\binom{n-1}{n/2}}, \tag{5.31}$$

and $|\phi\rangle$ is almost another eigenvector of U' with eigenvalue 1.

Shenvi, et al.[92] showed that in $\varOmega$, there are two orthonormal eigenvectors $|w_\pm\rangle$ of U', with corresponding eigenvalues $e^{\pm\mathrm{i}\theta}$ satisfying $\cos(\theta)>1-2/(3n)\approx 1$. It turns out that the two eigenvalues are the only eigenvalues of U' satisfying the inequality. In the 2-space spanned by $|w_\pm\rangle$, U' acts as a rotation of angle $\theta\in(0,\pi/2)$. In fact,

$$\theta=\frac{1}{c\sqrt{2^{n-1}}}+O\Big(\frac{n^{3/2}}{2^n}\Big)=O\Big(\frac{1}{\sqrt{N}}\Big)\ll 1. \tag{5.32}$$

So the 2-space spanned by $|\varpi_{N\times n}\rangle, |\phi\rangle$ almost agrees with the 2-space spanned by $|w_\pm\rangle$. According to the delicate estimations made in [92], the initial state $|\varpi_{N\times n}\rangle$ satisfies

$$|\varpi_{N\times n}\rangle=a(|w_+\rangle+|w_-\rangle)+O(\sqrt{n/N})\,|r\rangle, \tag{5.33}$$

where $0<a\leq 1$, and $|r\rangle\in\varOmega$ is a unit vector orthogonal to $|w_\pm\rangle$. After t iterations of U',

$$(U')^t|\varpi_{N\times n}\rangle=\cos(t\theta)|\varpi_{N\times n}\rangle+\sin(t\theta)|\phi\rangle+O(n^{3/4}/\sqrt{2^n}), \tag{5.34}$$

each iteration being an angle-θ rotation approximately from $|\varpi_{N\times n}\rangle$ towards $|\phi\rangle$.

In conclusion, starting from $|\varpi_{N\times n}\rangle$, it suffices to reach $|\phi\rangle$ with high precision by $t=\pi/(2\theta)=O(\sqrt{2^n})$ iterations of U'. The setup stage in Algorithm 5 has step complexity $O(\log N)$. In the checking stage, the flip operation has step complexity 1. So the total step complexity of the algorithm is $O(\sqrt{N})$.

In 2008, Potocek, et al.[257] showed that if the first register is divided into two subspaces for even and odd elements respectively, then they can evolve separately by the shift operator. In this way the probability of finding the solution doubles to almost 1. In 2015, Tonchev[258] obtained the same result by using a different coin transformation.

5.5 Quantization of Random Walk

The quantization of a random walk was defined by Szegedy[95] in 2004, and further refined in [96]. In this paper, the given Markov chain (Ω, P) is always assumed to be irreducible, and $N = |\Omega|$.

The state space of the quantization of (Ω, P) is $\mathbb{C}^N \otimes \mathbb{C}^N$, whose basis states are the following "directed edges" in Ω:

$$\Omega_{\mathrm{Q}} := \{|u, v\rangle \bigm| u, v \in \Omega,\, p_{uv} \neq 0\}. \tag{5.35}$$

Thus, the walker is on a graph whose vertices are the directed edges of Ω defined by the nonzero entries of P. Alternatively, one can take the coin space to be a copy of the position space.

The evolution is the composition of two unitary operators. The first is the flip operation F controlled by the position state, so that if the current position state is $|x\rangle$, and the coin state is the superposition of all $|y\rangle$ with probability p_{xy} and amplitude $\sqrt{p_{xy}}$, then the coin state is invariant under the flip operation, because it is exactly the probability distribution of the next step that the classical random walker P makes, after leaving position $|x\rangle$.

For all $x \in \Omega$, denote

$$|p_x\rangle := \sum_{y \in \Omega} \sqrt{p_{xy}}|y\rangle. \tag{5.36}$$

The $|x, p_x\rangle$ for all $x \in \Omega$ span an N-space of $\mathbb{C}^N \otimes \mathbb{C}^N$, denoted by

$$\mathcal{A} := \mathrm{span}\,\{|x, p_x\rangle \bigm| x \in \Omega\}. \tag{5.37}$$

The reflection $R_{\mathcal{A}}$ with respect to $\mathcal{A}$ can be selected as the flip operation F.

The second is the shift operation S controlled by the coin state, so that if the current coin state is $|x\rangle$, and the position state is the superposition of all $|y\rangle$ with probability $p^\star_{xy}$ and amplitude $\sqrt{p^\star_{xy}}$, then the position state is invariant under the shift operation, because it is exactly the probability distribution of the next step that the classical time-reversed random walker $P^\star$ makes, after leaving position $|x\rangle$.

Denote

$$|p^\star_x\rangle := \sum_{y \in \Omega} \sqrt{p^\star_{xy}}|y\rangle, \tag{5.38}$$

for all $x \in \Omega$. The $|p^\star_x, x\rangle$ span another N-space of $\mathbb{C}^N \otimes \mathbb{C}^N$, denoted by

$$\mathcal{B} := \mathrm{span}\,\{|p^\star_x, x\rangle \bigm| x \in \Omega\}. \tag{5.39}$$

The reflection $R_{\mathcal{B}}$ with respect to $\mathcal{B}$ can be selected as the shift operation S.

The two N-subspaces $\mathcal{A}, \mathcal{B}$ in the N^2-dimensional state space $\mathbb{C}^N \otimes \mathbb{C}^N$ have nontrivial intersection. For example,

$$|\pi\rangle := \sum_{u \in \Omega} \sqrt{\pi_u}|u, p_u\rangle = \sum_{v \in \Omega} \sqrt{\pi_v}|p^\star_v, v\rangle \tag{5.40}$$

is in $\mathcal{A} \cap \mathcal{B}$.

The evolution operator is denoted by

$$W(P) := R_{\mathcal{B}} R_{\mathcal{A}}. \tag{5.41}$$

It is a rotation in the state space. $(\mathrm{span}(\Omega_{\mathrm{Q}}), W(P))$ is called the *quantization* of random walk (Ω, P), or more accurately, of random walk (Ω, P) in the coin space followed by the time-reversed random walk $(\Omega, P^\star)$ in the position space.

Define matrix

$$P^{\surd} := (\sqrt{p_{ij}})_{i,j=0,1,\cdots,N-1}. \tag{5.42}$$

Then its rows have norm 1, and itself has norm $\sqrt{N}$. In the notation of (4.15),

$$|P_i^{\surd}\rangle = \sum_{j=0}^{N-1} \sqrt{p_{ij}}|j\rangle = |p_i\rangle, \qquad |P_{\|}^{\surd}\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle = |\varpi_N\rangle. \tag{5.43}$$

So $\mathcal{A}$ is the image space $\mathcal{V}_{\mathcal{M}}$ of the linear map $L_{\mathcal{M}}$ defined by (4.16), and $R_{\mathcal{A}} = R_{\mathcal{V}_{\mathcal{M}}}$ in (4.18).

According to [113], the implementation of $R_{\mathcal{A}}$ based on the unitary operator $U_{\mathcal{M}}$ in (4.17) has time complexity $O(\mathrm{poly}\log N)$. If furthermore (Ω, P) is reversible, then in (5.38), $|p_i^\star\rangle = |p_i\rangle$ for all $0 \le i < N$, so by (5.39), $R_{\mathcal{B}}$ can also be implemented efficiently. Hence, $W(P)$ can be efficiently implemented for all reversible (Ω, P).

Recall that in (4.19), a unitary $W = R_{\mathcal{V}_{\mathcal{M}}} R_{\mathcal{V}_{\mathcal{N}}}$ is defined, with the property that when

$$\{\sigma_k = \|A\| \cos(\alpha_k),\ k = 0, 1, \cdots, N-1\} \tag{5.44}$$

are all the singular values of matrix A counting multiplicity, where $\alpha_k \in [0, \pi/2)$, then (according to (4.23) and (4.25)) a portion of the spectrum of W is

$$\{\mathrm{e}^{\pm 2\mathrm{i}\alpha_k},\ k = 0, 1, \cdots, N-1\}. \tag{5.45}$$

This relation lies the foundation of Algorithm 4 for quantum SVE.

In the current setting, matrix $A = P^{\surd}$, and "SVE unitary" $W = R_{\mathcal{A}} R_{\varpi_N}$, where R_{ϖ_N} is the Grover diffusion. Notice the similarity between W and the quantum walk matrix $W(P) = R_{\mathcal{B}} R_{\mathcal{A}}$. The spectrum of $W(P)$ should be related to the singular values of matrix P, in a similar way as (5.45) to (5.44). This relation is disclosed by the following theorem of Szegedy[95].

Theorem 5.1 *Let P be an irreducible Markov chain, and let $\cos\theta_1, \cdots, \cos\theta_m$ be the multiset of singular values of $D(P)$ that lie in the open interval $(0, 1)$, where $\theta_i \in (0, \pi/2)$. Then the complete spectrum of $W(P)$ is:*

- *On $\mathcal{A} + \mathcal{B}$ those eigenvalues of $W(P)$ that have nonzero imaginary part are exactly $\mathrm{e}^{\pm 2\mathrm{i}\theta_1}, \cdots, \mathrm{e}^{\pm 2\mathrm{i}\theta_m}$, with the same multiplicity.*
- *On $\mathcal{A} \cap \mathcal{B}$ the operator $W(P)$ acts as the identity I. $\mathcal{A} \cap \mathcal{B}$ is spanned by the left (and right) singular vectors of $D(P)$ with singular value 1.*
- *On $\mathcal{A} \cap \mathcal{B}^\perp$ and $\mathcal{A}^\perp \cap \mathcal{B}$ the operator $W(P)$ acts as $-I$. $\mathcal{A} \cap \mathcal{B}^\perp$ (resp. $\mathcal{A}^\perp \cap \mathcal{B}$) is spanned by the left (resp. right) singular vectors of $D(P)$ with singular value 0.*

- *On $\mathcal{A}^{\perp} \cap \mathcal{B}^{\perp}$ the operator $W(P)$ acts as I.*

If (Ω, P) is reversible, i.e., $D(P)$ is symmetric, then the singular values of $D(P)$ are equal to the absolute values of the eigenvalues of P.

Corollary 5.2 (see [96]) *Let P be an ergodic and reversible Markov chain. On $\mathcal{A}+\mathcal{B}$ the spectrum of $W(P)$ is:*

- *$|\pi\rangle$ of (5.40) is the unique eigenvector corresponding to eigenvalue 1.*
- *$\mathrm{e}^{\pm 2\mathrm{i}\theta_j}$ are eigenvalues, for all singular value $\cos\theta_j \in (0,1)$ of $D(P)$.*
- *All the remaining eigenvalues are -1.*

Given a subset $\mathcal{M} \subset \Omega$, for the absorbing walk $(\Omega, P_{\mathcal{M}})$ of $\mathcal{M}$ defined by (5.12), there is a corresponding quantization $W(P_{\mathcal{M}})$. The *quantum hitting time*[95] h_{Q} of $\mathcal{M}$ refers to the smallest $t \in \mathbb{N}$ for which

$$\left\|(W(P_{\mathcal{M}}))^t|\pi\rangle - |\pi\rangle\right\| \geq 1/10. \tag{5.46}$$

6 Frameworks of Search: From Classical to Quantum Algorithms

In a finite set of states Ω, assume that a subset $\mathcal{M}$ of states are marked. Given a procedure C that, on input $u \in \Omega$ and an associated data structure $d(u)$, checks whether the state u is marked, the goal of search is either to find an element of $\mathcal{M}$ when promised that $\mathcal{M} \neq \emptyset$ (called the *finding problem*), or to determine whether $\mathcal{M}$ is nonempty (called the *decision problem*).

A typical search algorithm progresses in three stages[210]:

- In the setup stage, access some state of Ω (usually a random state).
- In the walk stage, move from state to state by performing a random walk or quantum walk. The moves are called updates.
- In addition, perform checking in the walk stage to see if the current state is marked.

The algorithm is required to be correct with probability at least 2/3 in either case, the finding or the decision problem.

The data structure (or data function) d stores some information of its argument, based on which it can be determined whether the argument is marked or not. Creating and maintaining the data structure incurs certain cost. The query complexity refers to the number of visits to procedure C that determines whether an element is marked or not. In search algorithms, the cost refers to the query complexity by default.

Corresponding to the three stages, there are three costs in a search algorithm[95]:

- **Setup cost** $\mathtt{S}$ In classical algorithms, $\mathtt{S}$ is the cost to sample from Ω and to construct $d(u)$ from sample u. In quantum algorithms, $\mathtt{S}$ is the cost to create a state that is the superposition of all the states in the search space.

- **Update cost** U In classical algorithms, U is the cost to simulate a transition in Ω from u to v, and to update $d(u)$ to $d(v)$. In quantum algorithms, U is the cost to create a state that is a superposition of all the states adjacent to the current state.
- **Checking cost** C The cost of checking if $u \in \mathcal{M}$ by information $d(u)$.

A naïve algorithm to solve the search problem is to repeat the sampling from Ω uniformly and checking the sample until a marked element is found. The quantum analogue of this naïve algorithm is Grover search.

6.1 Grover Search Revisited

Let $N = 2^n$ and $\Omega = \mathbb{Z}_N$, let f be a map from Ω to $\mathbb{Z}_2$, and assume that there is a unique element $x_0 \in \mathbb{Z}_N$ such that $f(x) = 1$ if and only if $x = x_0$. The search problem is to find x_0.

In the quantum setting, the basis states are

$$\{|x\rangle \mid x = 0, 1, \cdots, N-1\}. \tag{6.1}$$

Map f is taken as an oracle. Define unitary operator

$$\mathcal{O}_f : |x\rangle \mapsto (-1)^{f(x)}|x\rangle, \quad \forall x \in \Omega. \tag{6.2}$$

It is the reflection in space $\mathbb{C}^N$ with respect to hyperspace $|x_0\rangle^\perp$.

Call $|x_0\rangle$ the "good" space, and hyperspace $|x_0\rangle^\perp$ the "bad" space. Suppose the initial state is $|\psi\rangle$. Then

$$|\psi\rangle = a|\psi_{\text{good}}\rangle + |\psi_{\text{bad}}\rangle, \tag{6.3}$$

where $|\psi_{\text{good}}\rangle = |x_0\rangle$, and $|\psi_{\text{bad}}\rangle$ is the component in the bad space, and $|a| \leq 1$.

If the initial state $|\psi\rangle$ is chosen such that $|a| = 1$, then $|x_0\rangle$, and hence x_0, is found. If $|a| < 1$, suppose an appropriate initial state $|\psi\rangle$ is selected such that $0 < a < 1$, then $a = \cos(\theta/2)$ for some $0 < \theta < \pi$, and $\theta/2$ is the angle between vectors $|\psi\rangle$ and $|\psi_{\text{bad}}\rangle$, as shown in Figure 1.

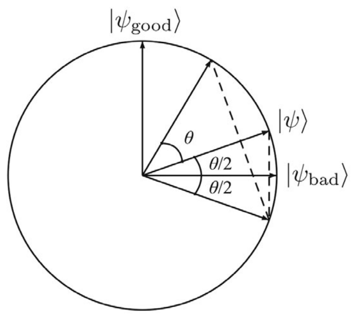

Figure 1 Geometric interpretation of Grover iteration

When $\theta \ll 1$, Grover's idea[38] is to rotate vector $|\psi\rangle$ in the plane spanned by $|\psi_{\text{good}}\rangle$ and $|\psi_{\text{bad}}\rangle$ towards vector $|\psi_{\text{good}}\rangle$ consistently, every time with fixed rotation angle θ, so that if θ

is small, then after some rounds of rotation towards $|\psi_{\mathrm{good}}\rangle$ (called *Grover iteration*), the angle θ' between the updated state, denoted by $|\psi'\rangle$, with $|\psi_{\mathrm{bad}}\rangle$ is almost $\pi/2$. By

$$|\psi'\rangle = \sin(\theta')|\psi_{\mathrm{good}}\rangle + \cos(\theta')|\psi_{\mathrm{bad}}\rangle, \tag{6.4}$$

the probability of obtaining $|\psi_{\mathrm{good}}\rangle$ by measurement is $\sin^2(\theta') \approx 1$, i.e., x_0 can be found with high probability by measuring $|\psi'\rangle$.

Every rotation in the plane is the composition of two reflections. Figure 1 shows that the angle-θ rotation towards $|\psi_{\mathrm{good}}\rangle$ is simply the reflection $\mathcal{O}_f$ followed by the reflection $R_{|\psi\rangle}$. Grover selected $|\varpi_N\rangle$ as the initial state $|\psi\rangle$. Then $R_{|\psi\rangle}$ is Grover's diffusion, and

$$|\psi_{\mathrm{bad}}\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq x_0} |x\rangle, \tag{6.5}$$

and the rotation angle $\theta \in (0, \pi)$ satisfies

$$\sin(\theta/2) = \langle \varpi_N | x_0 \rangle = 1/\sqrt{N}. \tag{6.6}$$

When N is sufficiently large, $\theta \approx 2/\sqrt{N}$.

After t times of Grover iteration, the angle between the updated state and $|\psi_{\mathrm{bad}}\rangle$ becomes

$$\theta_t := t\theta + \theta/2. \tag{6.7}$$

As $\sin^2(\theta_t)$ is the probability of observing $|\psi_{\mathrm{good}}\rangle$ in the updated state, to have $\sin^2(\theta_t) \approx 1$, i.e., $\theta_t \approx \pi/2$, it suffices to select $t \approx \frac{\pi\sqrt{N}}{4} - \frac{1}{2} = \Theta(\sqrt{N})$.

Algorithm 6 Grover search[38]

```
1: [Setup] Prepare the initial state
       $$|\varpi_N\rangle = \frac{1}{\sqrt{N}}|\psi_{\mathrm{good}}\rangle + \frac{\sqrt{N-1}}{\sqrt{N}}|\psi_{\mathrm{bad}}\rangle. \tag{6.8}$$
2: Do the following $O(\sqrt{N})$ times:
   2.1) [Checking] Apply $\mathcal{O}_f$ (reflection with respect to $|\psi_{\mathrm{good}}\rangle^{\perp}$).
   2.2) [Update] Apply $R_{\varpi_N}$ (reflection with respect to $|\varpi_N\rangle$).
3: Measure the final state.
```

Denote $\mathcal{M} = \{x_0\}$. Then $|\mathcal{M}| = 1$. Let ε be the proportion of marked items $\mathcal{M}$ in state space Ω. Then $\varepsilon = 1/N$.

If using the classical naïve algorithm for the search problem, then the setup subroutine is a sampling from Ω, the checking subroutine is a visit to f, and the update subroutine is another sampling from Ω. The query occurs only in the checking subroutine. The checking-and-update cycle needs to run $O(N)$ times to find x_0. The setup cost $\mathtt{S}$ and update cost $\mathtt{U}$ are both zero, and the checking cost $\mathtt{C} = O(N)$.

In Grover's algorithm, the number of visits to f is zero in both the setup and the update stages (so $\mathtt{S} = \mathtt{U} = 0$), and is $O(\sqrt{N})$ in the checking stage (so $\mathtt{C} = O(\sqrt{N})$). The total costs of the two algorithms are respectively:

$$\text{classical naïve}: \ \mathtt{S} + \frac{1}{\varepsilon}(\mathtt{U} + \mathtt{C}) = O(N); \qquad \text{Grover}: \ \mathtt{S} + \frac{1}{\sqrt{\varepsilon}}(\mathtt{U} + \mathtt{C}) = O(\sqrt{N}). \tag{6.9}$$

Grover's algorithm gains quadratic speedup over the naïve algorithm in parameter $1/\varepsilon$.

When $|\mathcal{M}| > 1$, then $\varepsilon = |\mathcal{M}|/N$. By modifying Step 2 of Grover's algorithm such that the checking-and-update cycle repeats $1/\sqrt{\varepsilon}$ times, the algorithm becomes valid for all $|\mathcal{M}| > 0$. It has query complexity $O(1/\sqrt{\varepsilon})$, and still achieves quadratic speedup over classical algorithms.

Remark 6.1 The following are some further extensions of Grover's algorithm.

1) Denote $k = |\mathcal{M}| > 0$. If k is not given beforehand, a quantum algorithm called *quantum counting* was proposed by Brassard, et al.[51] in 1998 to get an estimate $\widetilde{k}$ of k. If the relative error is τ, then the quantum counting algorithm has query complexity $O(1/(\tau\sqrt{\varepsilon}))$. This overhead does not increase the complexity of Grover's algorithm when $k = O(1)$.

2) Suppose there is a quantum algorithm $\mathcal{A}$ that uses no intermediate measurement (so that it remains a unitary operator), and has probability $p \ll 1$ of finding a marked state when applied to an initial state $|\phi\rangle$. Now revise Grover's algorithm as follows: Change the update unitary operation to $\mathcal{A}R_{|\phi\rangle}\mathcal{A}^{-1}$, and change the number of checking-and-update cycles to $O(1/\sqrt{p})$. Then with high probability a marked state can be found by the revised Grover's algorithm. The revision is well known as the *amplitude amplification algorithm*[52].

3) The time complexity of Grover's algorithm is $\widetilde{O}(1/\sqrt{\varepsilon})$. Suppose that f is no longer an oracle, and the time cost for evaluating $f(x)$ at $x = i$ is t_i for $0 \le i < N$. In 2010, Ambainis[259] proposed an algorithm that solves the problem in time $\widetilde{O}(\sqrt{\sum_{i=0}^{N-1} t_i^2})$, and proved that this time complexity is optimal.

4) If all the k elements $i \in \Omega$ such that $f(i) = 1$ are required to be found, by finding them one by one, the query complexity is

$$O\big(\sqrt{N/k} + \sqrt{(N-1)/(k-1)} + \cdots + \sqrt{(N-k+1)/1}\,\big) = O(\sqrt{kN}). \tag{6.10}$$

5) Grover's algorithm can be modified to determine if $\mathcal{M}$ is empty, and the modified version is called the *randomized Grover's algorithm*[96]. Let $\varepsilon = |\mathcal{M}|/N$. First sample from Ω a few times to accommodate the case that $\varepsilon \ge 1/4$. If no marked element is found, then proceed as if $\varepsilon < 1/4$. Choose an integer T uniformly at random in $[0, \sqrt{N}]$, make Grover iteration T times to the initial state in Grover's algorithm, and measure the final state. If $\mathcal{M}$ is not empty, a marked state is found by measurement with probability $\ge 1/4$ (Lemma 2 in [260]), otherwise no marked element can be found.

6) The Grover iteration is cyclic, so the knowledge of k is necessary to stop it at the right time to find a solution. To go beyond this restriction, in 2005 Grover proposed a fixed-point search algorithm[261], which converges monotonously to the good state, but no longer has quadratic speedup. In 2014, Yoder, et al.[262] proposed a bounded-error quantum search

algorithm that varies the phase shift as a function of the iteration number, and can achieve quadratic speedup.

6.2 Application: Collision Problem

A collision in a function f defined on $\{1, 2, \cdots, n\}$, is composed of two distinct elements $i, j \in \{1, 2, \cdots, n\}$ such that $f(i) = f(j)$. Suppose function f is r-to-one (every image has exactly $r > 0$ pre-images). Find a collision.

The problem is of particular interest to cryptology. Some functions such as hash functions are used in various cryptographic protocols, and the security of these protocols depends crucially on the presumed difficulty of finding collisions in such functions. When f is two-to-one, the most efficient classical algorithms require $\Theta(\sqrt{n})$ evaluations of f.

In 1997, Brassard, et al.[263] proposed a quantum algorithm with query complexity $O(\sqrt[3]{n/r})$, which is also the complexity lower bound[54]. This algorithm is based on a smart use of Grover search, and is presented below.

Algorithm 7 Brassard, et al.'s algorithm for function collision problem[263]

```
1: Choose an arbitrary subset $K \subset \{1, 2, \cdots, n\}$ of cardinality $k$.
2: Construct a table $L$ where each item holds a distinct pair $(i, f(i))$ with $i \in K$, then sort $L$
   according to the second entry in each item.
3: If $L$ contains a collision, stop and output the collision.
4: Search for an index $j \notin K$ such that there exists $i \in K$ with $f(j) = f(i)$.
```

Complexity analysis. The algorithm requires k queries in the first three steps. In the last step, since K has no collision, in space $\Omega = \{1, 2, \cdots, n\} \setminus K$, the cardinality of the following marked set

$$\mathcal{M} = \{j \in \Omega \,|\, \exists i \in K, \text{ s.t. } f(j) = f(i)\} \tag{6.11}$$

is $(r-1)k$, so Grover search needs $O\big(\sqrt{\frac{n-k}{(r-1)k}}\big)$ queries. The total number of queries is $O(k) + O\big(\sqrt{\frac{n-k}{(r-1)k}}\big)$. Let $k = O((n/r)^x)$, where $0 < x < 1$ is an unknown constant. Since $r > 1$, $1/(r-1) \leq 2/r$. Then

$$O(k) + O\left(\sqrt{\frac{n-k}{(r-1)k}}\right) = O\left((n/r)^x\right) + O\left((n/r)^{\frac{1-x}{2}}\right), \tag{6.12}$$

which takes the minimum $O((n/r)^{1/3})$ when $x = \frac{1-x}{2}$.

Besides the function collision problem, there is also the graph collision problem. Given an undirected graph of n vertices and an oracle access to a labeling of these vertices by 1 or 0, the graph collision problem is to detect if there are two neighboring vertices that are both labeled by 1, and if so, find such a couple. The best complexity lower bound is $\Omega(n^{1/2})$. In 2007, Magniez, et al.[60] gave an $O(n^{2/3})$-query quantum algorithm for collision search on general graphs. This still remains the best result.

6.3 MNRS Quantum Walk Framework

In classical search, random walk can be used to improve the efficiency. Algorithm 8 below follows this idea. It has complexity $\mathtt{S}+(\mathtt{U}+\mathtt{C})/(\varepsilon\delta)$, where δ is the spectral gap of the transition matrix P, and

$$\varepsilon = \sum_{u\in\mathcal{M}} \pi_u, \tag{6.13}$$

where $\boldsymbol{\pi}^{\mathrm{T}} = (\pi_u)_{u\in\Omega}$ is the stationary distribution of the random walk. If $\boldsymbol{\pi}^{\mathrm{T}}$ is the uniform distribution, then $\varepsilon = |\mathcal{M}|/|\Omega|$ is the proportion of marked items.

Algorithm 8 Classical search by one-step random walk

```
1: [Setup] Sample from $\Omega$ to get a sample $u$.
2: Do the following $h = O(1/(\varepsilon\delta))$ times, according to estimation (5.11):
   2.1) [Checking] If $u$ is marked, then return $u$ and stop.
   2.2) [Update] Start from $u$, simulate random walk $P$ on $\Omega$ for one time step.
3: Return "no marked item".
```

In Substep 2.2 of Algorithm 8, instead of simulating the random walk for once, one can make the simulation several times, i.e., the walker can go several steps instead of just one step, with the purpose of improving the efficiency of the search. This idea is embodied in the following algorithm, which has complexity:

$$\mathtt{S} + \frac{1}{\varepsilon}(\frac{1}{\delta}\mathtt{U} + \mathtt{C}). \tag{6.14}$$

Algorithm 9 Classical search by multi-step random walk[96, 250]

```
1: [Setup] Sample from $\Omega$ to get a sample $u$.
2: Do the following $h(\boldsymbol{\pi}^{\mathrm{T}}, \mathcal{M}) = O(1/\varepsilon)$ times (it is the average hitting time starting from the
   stationary distribution on $\Omega$):
   2.1) [Checking] If $u$ is marked, then return $u$ and stop.
   2.2) [Update] Start from $u$, simulate random walk $P$ on $\Omega$ for $T_{\text{mix}} = O(1/\delta)$ time steps (to
        reach the stationary distribution $\boldsymbol{\pi}^{\mathrm{T}}$).
3: Return "no marked item".
```

The quantum analogues of Algorithm 8 include Szegedy's quantum walk algorithm[95], Falk's spatial search algorithm[264], etc. The quantum analogues of Algorithm 9 include MNRS quantum walk algorithm[96], nested quantum walks, etc.

Below we introduce MNRS quantum walk algorithm, one of the most often used quantum walk frameworks. In [96], the following theorem was proved: Let $P = (p_{uv})$ be an ergodic and reversible Markov chain on Ω with spectral gap δ, and let $\varepsilon > 0$ be a lower bound of the probability that an element chosen from the stationary distribution π of P is marked. If $\mathtt{C}$ is the cost to check if a state $u \in \Omega$ is marked using data information $|d(u)\rangle$, $\mathtt{S}$ is the cost to construct the superposition $\sum_{u\in\Omega}\sqrt{\pi(u)}|u, d(u)\rangle$, and $\mathtt{U}$ is the cost to construct the superposition $\sum_{v\in\Omega}\sqrt{p_{uv}}|v,d(v)\rangle$ for any $u\in\Omega$, then a marked state can be found with high probability in the cost of

$$\mathtt{S}+\frac{1}{\sqrt{\varepsilon}}\left(\frac{1}{\sqrt{\delta}}\mathtt{U}+\mathtt{C}\right). \tag{6.15}$$

The algorithm gains quadratic speedup over Algorithm 9 in parameters $1/\varepsilon$ and $1/\delta$.

We investigate MNRS quantum walk algorithm with some detail, with the purpose to show that in (6.15), the quadratic speedups in $1/\varepsilon$ and $1/\delta$ come from Grover search and quantum phase estimation respectively.

Consider the simple case when $|\mathcal{M}|=1$, $|\Omega|=N$, and there is no need to construct data function d on Ω. Since $P=P^\star$, (5.38) becomes

$$|p_x^\star\rangle=\sum_{y\in\Omega}\sqrt{p_{xy}}|y\rangle=|p_x\rangle. \tag{6.16}$$

Similar to (6.5), define

$$|\pi_{\text{good}}\rangle=\frac{1}{\sqrt{\sum_{u\in\mathcal{M}}\pi_u}}\sum_{u\in\mathcal{M}}\sqrt{\pi_u}|u\rangle|p_u\rangle,\qquad |\pi_{\text{bad}}\rangle=\frac{1}{\sqrt{\sum_{u\notin\mathcal{M}}\pi_u}}\sum_{u\notin\mathcal{M}}\sqrt{\pi_u}|u\rangle|p_u\rangle. \tag{6.17}$$

The MNRS quantum walk algorithm has the same structure with Grover's algorithm:

Algorithm 10 Search by MNRS quantum walk framework[96]

```
1: [Setup] Prepare the initial state (5.40):
   $$|\pi\rangle=\sum_{u\in\Omega}\sqrt{\pi_u}|u\rangle|p_u\rangle=\sqrt{\sum_{u\in\mathcal{M}}\pi_u}\,|\pi_{\text{good}}\rangle+\sqrt{\sum_{u\notin\mathcal{M}}\pi_u}\,|\pi_{\text{bad}}\rangle. \tag{6.18}$$
2: Do the following $O(1/\sqrt{\varepsilon})$ times:
   2.1) [Checking] Make reflection $R_{|\pi_{\text{bad}}\rangle}$ with respect to $|\pi_{\text{bad}}\rangle$.
   2.2) [Update] Make reflection $R_{|\pi\rangle}$ with respect to $|\pi\rangle$.
3: Measure the final state.
```

Reflection $R_{|\pi_{\text{bad}}\rangle}$ has the effect that for any basis state in (6.18), if the first register is marked, then a minus sign is added to the coefficient. This can be done by an oracle similar to $\mathcal{O}_f$ in Grover's algorithm. Realizing reflection $R_{|\pi\rangle}$ is based on quantum eigenphase decomposition of unitary operator $W(P)$ (quantization of random walk P), which needs some elaboration.

Since (Ω,P) is reversible, matrix P has only real eigenvalues, all of which are in interval $[-1,1]$. Let the multiset of singular values of $D(P)$ in the interval $(0,1)$ be $\{\cos(\theta_j),\ j=1,\cdots,m\}$, where $\theta_j\in(0,\pi/2)$.

For the two N-spaces $\mathcal{A},\mathcal{B}$ defined by (5.37), (5.39) respectively, let the dimension of $\mathcal{A}+\mathcal{B}$ be n. By Theorem 5.1 and Corollary 5.2, when $W(P)=R_{\mathcal{B}}R_{\mathcal{A}}$ is restricted to the subspace $\mathcal{A}+\mathcal{B}$ of $\mathbb{C}^N\otimes\mathbb{C}^N$, its eigenvalues are

$$1;\quad\{\mathrm{e}^{2\mathrm{i}\theta_j},\ j=1,\cdots,m\};\quad\{\mathrm{e}^{-2\mathrm{i}\theta_j},\ j=1,\cdots,m\};\quad -1,\cdots,-1. \tag{6.19}$$

Furthermore, $|\pi\rangle$ is the eigenvector corresponding to eigenvalue 1.

As the spectral gap of P is δ, for any $1 \le j \le m$, $\delta \le 1 - \cos\theta_j$. By this and $\cos\theta_j > 1 - \theta_j^2/2$ for $\theta_j \in (0, \pi/2)$, one gets

$$\theta_j \ge \sqrt{2\delta}, \quad \forall 1 \le j \le m. \tag{6.20}$$

Let $\{\boldsymbol{w}_k,\ k = 0, \cdots, n-1\}$ be unit eigenvectors of $W(P)$ corresponding to the eigenvalues in the order of (6.19). Then $|\boldsymbol{w}_0\rangle = |\pi\rangle$. Set $\theta_0 := 0$, and $\theta_{m+j} := -\theta_j$ for $1 \le j \le m$. The eigenvalue corresponding to $\boldsymbol{w}_k$ for $k > 2m$ is -1. Set $\theta_k := \pi/2$ for $2m < k < n$. In this way, for all $0 \le k < n$, the eigenvalue of $W(P)$ corresponding to eigenvector $\boldsymbol{w}_k$ is $\mathrm{e}^{2\mathrm{i}\theta_k}$, where $-\pi/2 < \theta_k \le \pi/2$.

The eigenphase decomposition of $W(P)$ gives: For all $0 \le k < n$,

$$|\boldsymbol{w}_k\rangle|0\rangle \mapsto |\boldsymbol{w}_k\rangle|\widetilde{\theta}_k\rangle \mapsto (-1)^{\left[|\widetilde{\theta}_k| \le \sqrt{\delta}/2\right]}|\boldsymbol{w}_k\rangle|\widetilde{\theta}_k\rangle \mapsto (-1)^{\left[|\widetilde{\theta}_k| \le \sqrt{\delta}/2\right]}|\boldsymbol{w}_k\rangle|0\rangle, \tag{6.21}$$

where $\widetilde{\theta}_k$ is a $\sqrt{\delta}/2$-approximate of θ_k, and $L \mapsto [L]$ is a real-valued function in logic variable L such that $[L] = 0$ if L is true, and $[L] = 1$ if L is false.

(6.21) is an implementation of the reflection $R_{|\pi\rangle}$ in space $\mathcal{A} + \mathcal{B}$, because state $|\pi\rangle$ is invariant, while any state in the orthogonal complement of $|\pi\rangle$ is reversed, due to

$$|\widetilde{\theta}_k| \ge \left(\sqrt{2} - \frac{1}{2}\right)\sqrt{\delta} > \frac{1}{2}\sqrt{\delta}, \quad \forall k \ne 0. \tag{6.22}$$

By Proposition 3.1, the complexity of (6.21) is $O(\mathtt{U}/\sqrt{\delta})$, where $\mathtt{U}$ is the cost to implement $W(P)$ by preparing $R_{\mathcal{A}}, R_{\mathcal{B}}$ respectively. In the phase estimation procedure (3.2), quantum walk $W(P)$ is executed $O(1/\sqrt{\delta})$ times, which is determined by the phase estimation precision.

MNRS quantum walk algorithm is also valid when $|\mathcal{M}| > 0$, but it is valid only when $|\mathcal{M}|$ is known in advance. This assumption can be removed by some standard techniques, without increasing the asymptotic complexity of the algorithm[260].

Nested quantum walk.

In the checking subroutine or update subroutine of a quantum walk algorithm, another graph can be constructed on which a new quantum walk can be made. This is the *nested quantum walk framework*.

For example, let there be a nested MNRS quantum walk, where the inner MNRS quantum walk is in the checking subroutine, and has complexity

$$\mathtt{C} = \mathtt{S}' + \frac{1}{\sqrt{\varepsilon'}}\left(\frac{1}{\sqrt{\delta'}}\mathtt{U}' + \mathtt{C}'\right), \tag{6.23}$$

where $(\mathtt{S}', \mathtt{U}', \mathtt{C}', \varepsilon', \delta')$ are the costs and parameters associated with the inner walk. Then the nested MNRS quantum walk has complexity:

$$\mathtt{S} + \frac{1}{\sqrt{\varepsilon}}\left(\frac{1}{\sqrt{\delta}}\mathtt{U} + \mathtt{S}' + \frac{1}{\sqrt{\varepsilon'}}\left(\frac{1}{\sqrt{\delta'}}\mathtt{U}' + \mathtt{C}'\right)\right). \tag{6.24}$$

Jeffery, et al.[61] proposed a method to avoid repeated overhead of the setup cost, and decreased the complexity (6.24) to

$$\mathtt{S} + \mathtt{S}' + \frac{1}{\sqrt{\varepsilon}}\left(\frac{1}{\sqrt{\delta}}\mathtt{U} + \frac{1}{\sqrt{\varepsilon'}}\left(\frac{1}{\sqrt{\delta'}}\mathtt{U}' + \mathtt{C}'\right)\right). \tag{6.25}$$

6.4 Application: Element Distinctness Problem

Given a set of integers $\{x_1, \cdots, x_n\}$, the element distinctness problem is to detect if there exist two different indices i, j such that $x_i = x_j$, and if so, find such a pair. The best classical algorithm requires $O(n)$ queries, for example by sorting.

The first quantum algorithm, proposed by Buhrman, et al.[265] in 2005, achieves query complexity $O(n^{3/4})$ by using Grover's search in a clever two-level construction. In 2007, Ambainis[78] discovered a quantum algorithm with query complexity $O(n^{2/3})$ and time complexity $\widetilde{O}(n^{2/3})$. This algorithm is optimal in that each complexity reaches the lower bound[54].

Ambainis' algorithm is based on a symmetric walk in the Johnson graph $J(n, r)$. A vertex A is a subset of $\Omega = \{1, 2, \cdots, n\}$ with cardinality r. It is marked if there exist two different elements i, j of subset A such that $x_i = x_j$. The probability for an arbitrary vertex A to be marked is

$$\varepsilon \geq \mathrm{Prob}(i, j \in A) = \frac{\binom{n-2}{r-2}}{\binom{n}{r}} = \frac{r(r-1)}{n(n-1)} \approx \frac{r^2}{n^2}. \tag{6.26}$$

The data associated with vertex A is $d(A) := \{(u, x_u) : u \in A\}$.

Let A, A' be two neighboring vertices of $J(n, r)$. By definition, the two vertices when taken as subsets of cardinality r, have exactly $r - 1$ elements in common, so the random walk on $J(n, r)$ has transition matrix $P = (p_{A,A'})$, where $p_{A,A'} = \frac{1}{r(n-r)}$ if A, A' are neighbors. The spectral gap is $\delta \geq 1/r$ (see [250]).

In the MNRS quantum walk framework, the setup cost $\mathtt{S}$ is used to construct

$$\frac{1}{\sqrt{\binom{n}{r}}} \sum_{A \in J(n,r)} |A\rangle |d(A)\rangle, \tag{6.27}$$

so $\mathtt{S} = r$ queries. In the update subroutine, constructing $d(A')$ from $d(A)$ is done by querying the unique index in $A' \setminus A$ and "unquerying" the unique index in $A \setminus A'$. Since all other information of $d(A')$ are already in $d(A)$, $\mathtt{U} = 2$. In the checking subroutine, no additional query is needed, so $\mathtt{C} = 0$. The query complexity is thus $O(r + n/\sqrt{r})$, which achieves the minimum $O(n^{2/3})$ at $r = n^{2/3}$. Furthermore, an appropriate data structure for d can be used to make the time complexity equal to $\widetilde{O}(n^{2/3})$.

A generalization of element distinctness problem is element k-distinctness: To decide if there exist k different indices $i_1, \cdots, i_k$ of $\{1, 2, \cdots, n\}$ such that $x_{i_1} = \cdots = x_{i_k}$, and if so, find such a k-tuple. The lower bound of the query complexity is $\Omega(n^{2/3})$.

Ambainis' algorithm can be extended to solve the element k-distinctness problem, with query complexity $O(n^{k/(k+1)})$, and time complexity $\widetilde{O}(n^{k/(k+1)})$. In 2012, Belovs decreased the query complexity to $O(n^{0.75-1/(4(2^k-1))})$ by using learning graphs[266]. This remains the best query complexity result.

For the case $k = 3$, in 2013 Childs, et al.[97] proposed a quantum algorithm using $\widetilde{O}(n^{0.714})$ queries. For $k > 3$, in 2014 Jeffery[267] proposed a quantum algorithm that achieves currently the best time complexity: $\widetilde{O}(n^{(k-1)/k})$.

6.5 Other Frameworks of Quantum Walk Based Search

Szegedy's quantum walk framework is another important algorithm for the detection problem, and for the finding problem when $|\mathcal{M}| = 1$.

In the following algorithm for the detection problem, the absorbing quantum walk $W(P_{\mathcal{M}})$ serves as a process of both update and checking, so its cost is $\mathtt{U} + \mathtt{C}$. It is applied repeatedly to the initial state $|\pi\rangle$. If $\mathcal{M}$ is empty, then $P_{\mathcal{M}} = P$ and the initial state is left invariant. If $\mathcal{M}$ is nonempty, then the angle between vectors $(W(P_{\mathcal{M}}))^t|\pi\rangle$ and $(W(P))^t|\pi\rangle = |\pi\rangle$ gradually increases for t not too large[95].

Algorithm 11 Szegedy's quantum walk algorithm to detect if $|\mathcal{M}| = 0$[95]

```
1: [Setup] Prepare the initial state $|\pi\rangle$ as (5.40).
2: Apply $h_{\mathrm{Q}} = O(\sqrt{h})$ times the following walk:
   [Simultaneous Checking and Update]   $W(P_{\mathcal{M}})$.
3: Construct the following state (up to normalization), the first register being the control one:
   $$\frac{1}{2}|0\rangle\big(|\pi\rangle + (W(P_{\mathcal{M}}))^t|\pi\rangle\big) + \frac{1}{2}|1\rangle\big(|\pi\rangle - (W(P_{\mathcal{M}}))^t|\pi\rangle\big). \qquad (6.28)$$
4: Measure the final state (at the control register).
```

When (Ω, P) is ergodic and symmetric, then in the above algorithm, the quantum hitting time $h_{\mathrm{Q}} \leq \sqrt{h}$, where h is the classical "hitting time" (the time steps to determine if $\mathcal{M} = \emptyset$ by random walk (Ω, P) in classical search algorithms). So Algorithm 11 has complexity $\mathtt{S} + \sqrt{h}(\mathtt{U} + \mathtt{C})$.

If P is state-transitive and $|\mathcal{M}| = 1$, then after deleting Step 3 of Algorithm 11, the modified algorithm can be used to find the unique marked state with probability at least N/h with cost $\mathtt{S} + \sqrt{h}(\mathtt{U} + \mathtt{C})$. The success probability can be increased to $\Theta(1)$ with $\sqrt{h/N}$ iterations of the algorithm using quantum amplitude amplification.

In the framework of quantum walk, the finding problem for $|\mathcal{M}| > 1$ is generally more complicated than for $|\mathcal{M}| = 1$. For the finding problem with information given that there are multiple marked elements, in 2016, Krovi, et al.[268] introduced a novel idea called interpolating quantum walk; in 2017, Dohotaru and Høyer[269] developed a new technique called controlled quantum walk. None of them guarantees quadratic speedup over classical algorithms.

To deal with multiple marked elements, the concept of extended hitting time h^+ for classical random walk is important. In 2015, Ambainis and Kokainis[270] established the relation $h \leq h^+ \leq h/\delta$. In 2016, Krovi, et al.[268] proved $h = h^+$ if $|\mathcal{M}| = 1$, and $h \leq h^+$ if $|\mathcal{M}| > 1$. In 2017, Høyer and Komeili[253] established currently the best bound for h^+: $1/\varepsilon \leq h \leq h^+ \leq 1/(\varepsilon\delta)$.

Table 2 collects some typical frameworks of quantum walk based search.

Table 2 Frameworks for quantum walk based search and their costs

Algorithm	Cost for case $\lvert\mathcal{M}\rvert = 1$	Cost for case $\lvert\mathcal{M}\rvert > 1$
Szegedy[95] (2004)	$\sqrt{\frac{h}{N}}(\mathtt{S} + \sqrt{h}(\mathtt{U} + \mathtt{C}))$	—
MNRS[96] (2011)	$\mathtt{S} + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathtt{U} + \mathtt{C})$	$\mathtt{S} + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathtt{U} + \mathtt{C})$
Magniez, et al.[67] (2012)	$\mathtt{S} + \sqrt{h}(\mathtt{U} + \mathtt{C})$	—
Krovi, et al.[268] (2016)	$\mathtt{S} + \sqrt{h}(\mathtt{U} + \mathtt{C})$	$\mathtt{S} + \sqrt{h^+}(\mathtt{U} + \mathtt{C})$
Dohotaru and Høyer[269] (2017)	$\mathtt{S} + \sqrt{h}\,\mathtt{U} + \frac{1}{\sqrt{\varepsilon}}\mathtt{C}$	$\mathtt{S} + \sqrt{h^+}(\mathtt{U} + \mathtt{C})$

6.6 Application: Spatial Search

For a physical region in which the moving of a quantum robot from one location to an adjacent one takes unit time, the time needed to search the region could depend critically on its spatial layout. For example, if the items are arranged on a one-dimensional line, simply traveling from one end of the line to the other requires n moves, and no local algorithm, classical or quantum, can find a marked item in less time than $\Omega(n)$.

Consider a 2-dimensional grid of size $\sqrt{n} \times \sqrt{n}$, which has exactly one marked node. Direct application of Grover search gives step complexity $O(\sqrt{n} \times \sqrt{n}) = O(n)$, because during each of the $O(\sqrt{n})$ Grover iterations, the algorithm needs $O(\sqrt{n})$ steps to travel across the grid and return to its starting point for the diffusion step.

In 2003, by using Grover's algorithm recursively, Aaronson and Ambainis[68] gave an algorithm with time complexity $O(\sqrt{n}\log^{3/2} n)$. In 2004, Ambainis, et al.[65] presented an algorithm, based on discrete-time quantum walk, that finishes the search in $O(\sqrt{n}\log n)$ steps. In 2008, Tulsi[66] improved the result to $O(\sqrt{n\log n})$ by modifying the quantum walk. In 2013, Falk[264] used two kinds of quantum walks (diffusions) successively to reduce the complexity to $O(\sqrt{n})$, which is also the complexity lower bound. Below we introduce Falk's algorithm.

Falk's algorithm contains two key constructions and the corresponding diffusion operations. The first is a tessellation of the grid into 4×4 blocks, and the associated diffusion within each block. This diffusion takes local pieces of the grid and trades their amplitudes, so that if a particular piece of the grid contains the marked item, it will act like Grover search and start sending amplitudes to the marked node.

The second is another tessellation of the grid into 4×4 blocks, such that if two blocks of different tessellations overlap, the overlap is a quadrant of each block. The associated diffusion within a block B of the second tessellation disperses the amplitudes among the four blocks in the first tessellation that overlap with B.

For simplicity, assume $N = \sqrt{n}/4$ is an integer. The basis states are

$$\Omega = \{|i, j\rangle,\ \ 0 \leq i, j < 4N\}. \tag{6.29}$$

The first tessellation is composed of the following states:

$$|u_L(i, j)\rangle := \frac{1}{4}\sum_{x,y=0}^{3} |4i + x, 4j + y\rangle, \quad \forall i, j = 0, 1, \cdots, N - 1. \tag{6.30}$$

Denote

$$\mathcal{L} := \mathrm{span}\{|u_L(i,j)\rangle,\ i,j=0,1,\cdots,N-1\}. \tag{6.31}$$

The local diffusion operator associated with the tessellation is the reflection $R_{\mathcal{L}}$ with respect to $\mathcal{L}$.

The second tessellation is composed of the following states:

$$|u_A(i,j)\rangle := \frac{1}{4}\sum_{x,y=0}^{3} |4i+x+2, 4j+y+2\rangle. \tag{6.32}$$

Denote

$$\mathcal{A} := \mathrm{span}\{|u_A(i,j)\rangle,\ i,j=0,\cdots,N-1\}. \tag{6.33}$$

The amplitude dispersion operator associated with the tessellation is the reflection $R_{\mathcal{A}}$ with respect to $\mathcal{A}$.

Let $|i_0,j_0\rangle$ be the marked state. As in Grover search, an oracle (a mark-checking function) can be used to construct the reflection $R_{|i_0,j_0\rangle^\perp}$ with respect to the "bad" states. Falk's algorithm has similar structure with Grover's algorithm, with the exception that there are two walks/diffusions instead of one.

Algorithm 12 Search in a $\sqrt{n}\times\sqrt{n}$ grid for the unique marked node[264]

```
1: [Setup] Prepare the initial state
       $$|\varpi_{\sqrt{n}\times\sqrt{n}}\rangle = \frac{1}{\sqrt{n}}\sum_{i,j=0}^{4N-1}|i,j\rangle, \tag{6.34}$$
   by walking along a horizontal and then a vertical.
2: Apply the following operator $O(\sqrt{n})$ times:
   [Update 1, Checking, Update 2, Checking]   $R_{|i_0,j_0\rangle^\perp}R_{\mathcal{A}}R_{|i_0,j_0\rangle^\perp}R_{\mathcal{L}}$.
3: Measure the final state.
```

In the above algorithm, the setup cost $\mathtt{S}=O(\sqrt{n})$, the update and checking costs $\mathtt{U}+\mathtt{C}=O(1)$. The total complexity is

$$\mathtt{S}+\sqrt{n}(\mathtt{U}+\mathtt{C}) = O(\sqrt{n}). \tag{6.35}$$

Experiments[264] show that the actual probability of finding the marked node by the algorithm is approximately 1/2.

Inspired by Falk's work, in 2015 Portugal, et al.[271] presented a new framework of quantum walk — Staggered quantum walk model, and proved its equivalence with Szegedy's framework. In 2017, Høyer and Komeili[253] proposed a quantum walk based algorithm for finding multiple marked elements on the grid, which achieves quadratic speedup after ignoring logarithmic factor.

7 Some Open Problems and Emerging Directions

This section contains some famous open problems and new research directions in quantum algorithm design.

7.1 Lattice Problems

Lattice problems have important applications in coding theory and cryptographic systems due to their conjectured hardness. Since the invention of Shor's algorithm, a number of post-quantum public-key cryptosystems have been proposed, many of which are based on lattice problems. Lattice-based cryptosystems are important candidates for post-quantum cryptography, and are believed to be resistant to attacks from quantum computer.

Let $\{\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n\}$ be a basis of $\mathbb{R}^n$. The lattice generated by them is $L = \{\sum_{i=1}^n \lambda_i \boldsymbol{b}_i : \lambda_i \in \mathbb{Z}\}$. The shortest vector problem (**SVP**) is to find a nonzero vector of L with minimal L_2-norm. SVP is the most famous and well studied computational problem on lattices. Let $f(n)$ be a polynomial in n. The $f(n)$-unique-SVP is to find the shortest nonzero vector that is shorter by a factor of at least $f(n)$ than any other nonparallel vector. This problem also has important applications in cryptography[30, 31].

To solve SVP, in the classical computing domain, in 2014 Aggarwal, et al.[272] proposed an algorithm that has time complexity $2^{n+o(n)}$ and space complexity $2^{n+o(n)}$. In 2018, Chen, et al.[273] proposed another algorithm, with time complexity $2^{2.05n+o(n)}$ and space complexity $2^{0.5n+o(n)}$.

In the quantum computing domain, in 2013 Laarhoven, et al.[45] proposed a quantum algorithm based on Grover search. This algorithm has time complexity $2^{1.799n+o(n)}$ and space complexity $2^{1.286n+o(n)}$. In 2018, Chen, et al.[273] proposed a quantum algorithm with time complexity $2^{1.2553n+o(n)}$ and space complexity $2^{0.5n+o(n)}$.

Under certain heuristic assumptions, currently the best algorithm in the classical computing domain to solve SVP was proposed by Becker, et al.[274] in 2016. It has time complexity $2^{0.2925n+o(n)}$ and space complexity $2^{0.208n+o(n)}$.

Under the same heuristic assumptions but in the quantum computing domain, in 2013, Laarhoven, et al.[45] proposed a quantum algorithm with time complexity $2^{0.268n+o(n)}$ and space complexity $2^{0.268n+o(n)}$. In 2015, Laarhoven[275] in his Ph.D. dissertation further improved the result by decreasing both the time complexity and the space complexity to $2^{0.265n+o(n)}$.

7.2 HSP of Dihedral Groups (Dihedral HSP)

This problem is also one of the most important mathematical problems that seem to resist efficient quantum computing. An efficient algorithm for the HSP of dihedral groups would lead to an efficient algorithm for several cryptographically significant lattice problems[32].

Dihedral group $D_N := \{x, y \,|\, x^N = y^2 = yxyx = 1\}$ is the symmetry group of the regular N-gon in the plane, where x stands for the rotation with angle $2\pi/N$ about the center of the regular N-gon, and y stands for the reflection with respect to one diagonal. It is not hard to see that $D_N \cong \mathbb{Z}_N \rtimes \mathbb{Z}_2$, with multiplication $(a, r) * (b, s) := (a + (-1)^r b, r + s)$.

In 2000, Ettinger and Høyer[276] showed that the dihedral HSP can be reduced to the search of a subgroup of the form $H = \{(0, 0), (d, 1)\}$. So the dihedral HSP can be reformulated as follows: Let f be a map from $\mathbb{Z}_N \rtimes \mathbb{Z}_2$ to a finite set S, with the property that (1) $f(x, 0)$ is injective from $\mathbb{Z}_N$ to S, (2) there exists some $d \in \mathbb{Z}_N$ such that $f(x, 0) = f(x + d, 1)$ for all $x \in \mathbb{Z}_N$. Find d.

The dihedral HSP is closely related to the following *subset sum problem*: Given $m+1$ integers $z, y_1, \cdots, y_m \in \mathbb{Z}_N$, decide if there exist $s_1, \cdots, s_m \in \{0,1\}$ such that $\sum_{i=1}^m y_i s_i = z$. It is called the *average-case subset sum problem* if the equation is replaced by $\sum_{i=1}^m y_i s_i = z \bmod N$.

In 2004, Regev[32] proved that if there exists an efficient quantum algorithm to solve the average-case subset sum problem, then there exists an efficient quantum algorithm to solve the dihedral HSP. In 2006, Bacon, et al.[277] further specified that if the subset sum problem can be efficiently solved and its solutions be efficiently quantum sampled, then the pretty good measurement for the dihedral HSP can be efficiently implemented by quantum circuits, which when combined with the standard method for the dihedral HSP, would lead to an efficient quantum algorithm for solving the dihedral HSP.

The following procedure is known as the *standard method* for the dihedral HSP.

Step 1 Prepare the initial state

$$|\psi_1\rangle = \frac{1}{\sqrt{2N}} \sum_{x\in\mathbb{Z}_N, t\in\mathbb{Z}_2} |x\rangle|t\rangle|f(x,t)\rangle = \frac{1}{\sqrt{2N}} \sum_{x\in\mathbb{Z}_N} (|x\rangle|0\rangle + |x+d\rangle|1\rangle)|f(x,0)\rangle. \tag{7.1}$$

Step 2 Apply quantum Fourier transform to the first register. Result:

$$|\psi_2\rangle = \frac{1}{\sqrt{2N}} \sum_{x,y\in\mathbb{Z}_N} \omega^{xy}|y\rangle(|0\rangle + \omega^{dy}|1\rangle)|f(x,0)\rangle, \quad \text{where } \omega = \mathrm{e}^{2\pi \mathrm{i}/N}. \tag{7.2}$$

Step 3 Work on $|\psi_2\rangle$ to get d (open problem).

In 2004, Regev[32] showed that if there exists an efficient quantum algorithm based on the standard method for the dihedral HSP, then there exists an efficient quantum algorithm with the same complexity that solves $f(n)$-unique-SVP. Also in this year, Kuperberg[18] proposed a quantum algorithm to solve the dihedral HSP starting from the state (7.2), with time complexity and space complexity both $2^{O(\sqrt{\log N})}$. Later in the same year, Regev[278] improved the result by reducing the space complexity to $O(\mathrm{poly}\log N)$. This is still the best result for the problem.

7.3 Graph Isomorphism Problem

Suppose Γ_1 and Γ_2 are two undirected graphs on the same vertices $V = \{1, 2, \cdots, n\}$. The graph isomorphism problem concerns whether there exists a one-to-one map $\sigma : V \to V$ such that (i,j) is an edge of Γ_1 if and only if $(\sigma(i), \sigma(j))$ is an edge of Γ_2. If $\Gamma_1 = \Gamma_2$, such a σ is called an automorphism. The graph automorphism problem is to determine whether a graph has a non-trivial automorphism. The two problems have the same computational difficulty.

In the classical computing domain, currently the best algorithms for solving the graph isomorphism problem run in time $\mathrm{e}^{O(\sqrt{n\log n})}$ for general graphs[279] and $\mathrm{e}^{O(n^{1/3}(\log n)^2)}$ for strongly regular graphs[280]. In the quantum computing domain, the results are few and far between.

The graph isomorphism problem is related to the HSP of symmetric groups. Let Γ be a graph on n vertices, and define $f(\sigma) = \sigma(\Gamma)$ for all $\sigma \in S_n$. Then f is constant on every coset of the automorphism group of Γ. Since the automorphism group of Γ is a subgroup of S_n, the construction of f reduces the graph automorphism problem to an HSP of S_n[281–284]. An efficient algorithm for the HSP of the symmetric group would lead to an efficient algorithm for graph isomorphism problem.

For symmetric groups, the standard method for HSP is insufficient[285, 286], although circuit implementations of quantum Fourier transform on symmetric groups are available[283, 287, 288].

7.4 Triangle Finding

Let $G = (V, E)$ be a graph. A triangle of G is composed of three vertices that are neighbors to each other. Triangle finding is to detect if there is any triangle in a graph, and if so, find one. Let A be the adjacency matrix of G. A triangle exists if and only if matrix A^3 contains a nonzero diagonal entry.

Triangle finding plays a key role in many problems. It is proved that faster algorithms for triangle finding would lead to faster algorithms for matrix multiplication[81, 289], the 3-SUM problem[290], the Max-2SAT problem[291], etc.

Let n be the number of vertices of G. While classical algorithms have query cost $\binom{n}{3} = O(n^3)$, Grover search reduces the query cost to $O(n^{3/2})$. The first quantum algorithm outperforming Grover search for this problem was given by Magniez, et al.[60] in 2007, by considering a quantum walk on Johnson graph $J(n, r)$, and combining amplitude amplification with combinatorial arguments. This algorithm has query complexity $\widetilde{O}(n^{13/10})$.

In 2012, Belovs[292] used learning graph to decrease the query complexity to $O(n^{35/27}) \approx O(n^{1.296})$. In 2013, Lee, et al.[76] further decreased the query complexity to $O(n^{9/7}) \approx O(n^{1.285})$, again using learning graph. Both complexities can be achieved by nested quantum walks on two Johnson graphs. In 2014, Le Gall[75] proposed an algorithm with currently the best query complexity $O(n^{1.25})$, using a 4-level nested MNRS quantum walk.

A known lower bound of the quantum query complexity of this problem is $\Omega(n)$. It is not clear if the lower bound can be reached.

7.5 Quantum Optimization

Quantum optimization is an emerging research direction but still at its infancy. Two problems are concerned: Quantum semi-definite programming, and quantum convex optimization.

Quantum semi-definite programming.

Semi-definite program (**SDP**) is an important tool for designing efficient optimization and approximation algorithms[293]. SDP generalizes the better-known linear program (**LP**), and (like LP) is efficiently solvable.

The basic form of an SDP is the following: Given $n \times n$ Hermitian matrices $C, A_1, \cdots, A_m$, and real numbers $b_1, \cdots, b_m$, solve for

$$\text{Maximize}_{X=X_{n\times n}\succeq 0} \ \ \mathrm{Tr}(CX), \quad \text{subject to} \ \ \mathrm{Tr}(A_jX) \le b_j, \quad j = 1, 2, \cdots, m. \tag{7.3}$$

An LP corresponds to the case where all matrices are diagonal.

In the classical computing domain, currently the best SDP-solvers such as the one in [294], approximate the optimal value up to additive error ε with complexity:

$$O\big(m(m^2 + n^{\omega} + mns)\text{poly}\log(m, n, R, r, 1/\varepsilon)\big), \tag{7.4}$$

where $\omega \in [2, 2.373)$ is the optimal exponent for matrix multiplication; s is the maximal sparsity of the input matrices; R, r are respectively the upper bounds of the norm of the optimal primal and dual solutions of the SDP.

In the quantum computing domain, in 2017 Brandão and Svore[295] discovered a quantum algorithm that significantly outperforms classical SDP-solvers in certain problems. The complexity is $\widetilde{O}\big(\sqrt{mn}s^2(Rr/\varepsilon)^{32}\big)$, with speedup in parameters m, n, but speed-down in parameters $R, r, 1/\varepsilon$.

Later in the same year, Apeldoorn, et al.[296] provided a quantum algorithm with complexity $\widetilde{O}\big(\sqrt{mn}s^2(Rr/\varepsilon)^8\big)$. In 2018, Apeldoorn and Gilyén[297], Brandão, et al.[298] separately improved the result to

$$\widetilde{O}\big((\sqrt{m} + \sqrt{n}(Rr/\varepsilon))s(Rr/\varepsilon)^4\big). \tag{7.5}$$

The result is tight in the dependence on m, n, because there is a quantum lower bound[298] $\Omega(\sqrt{m} + \sqrt{n})$ when R, r, s, ε are constant.

In August 2018, for the case $m = O(n^2)$, Kerenidis and Prakash[242] used quantum interior point method to solve the SDP in time

$$\widetilde{O}\big(n^{3.5}\kappa^3\xi^{-2}\log(1/\varepsilon)\big), \tag{7.6}$$

where ξ, κ are respectively the error and maximal condition number of the Hessian matrices used in the method.

Quantum convex optimization.

The following is a general convex minimization problem: Given a convex domain $\mathcal{K} \subseteq R^n$ bounded by two balls $B(0, r)$ and $B(0, R)$, and given a bounded convex function $f : \mathcal{K} \to \mathbb{R}$ whose upper bound and lower bound are both known, solve for

$$\text{argmin}_{x\in\mathcal{K}}\ f(x). \tag{7.7}$$

In the classical computing domain, the lower bounds of the query complexity are: $\Omega(n)$ to the evaluation oracle for f, and $\Omega(n)$ to the membership oracle for $\mathcal{K}$. Currently the best algorithms such as the one in [299], solve the problem with $\widetilde{O}(n^2)$ queries of the evaluation oracle for f, and $\widetilde{O}(n^2)$ queries of the membership oracle for $\mathcal{K}$.

In the quantum computing domain, in September 2018, Chakraborty, et al.[300] showed that there is a quantum algorithm to solve the problem by $\widetilde{O}(n)$ queries of the evaluation oracle for f, and $\widetilde{O}(n)$ queries of the membership oracle for $\mathcal{K}$. They proved that the lower bounds of the query complexity are: $\Omega(\sqrt{n}/\log n)$ to the evaluation oracle for f, and $\Omega(\sqrt{n})$ to the membership oracle for $\mathcal{K}$. Similar results are obtained by Apeldoorn, et al.[301] in the same month of the year.

7.6 Quantum Algorithms with Limitation on Quantum Computing Resources

The current status of quantum computer hardware is that there are only a few qubits available for running a quantum algorithm. Designing quantum algorithms that outperform classical algorithms with the fewest resources is an important task.

Take as an example the resources used by Grover's algorithm in attacking cryptosystems. In 2016, Grass, et al.[302] estimated the number of qubits n_B, the number of elementary gates n_G, and the depth of circuits d_C, to run Grover's algorithm to attack AES. The resources for attacking AES-128, AES-192, AES-256 are respectively:

$$(n_B, n_G, d_C) \quad = \quad (2953,\ 2^{86},\ 2^{81});\ \ (4449,\ 2^{119},\ 2^{113});\ \ (6681,\ 2^{151},\ 2^{145}). \tag{7.8}$$

Due to the large circuit depth, it is challenging to implement this algorithm on an actual physical quantum computer to break AES.

In the same year, Amy, et al.[303] also estimated the resources to run Grover's algorithm to attack SHA-2 and SHA-3. They assumed that the pre-image attack is run on a surface code based fault-tolerant quantum computer. The resources for attacking SHA2-256 and SHA3-256 are respectively:

$$(d_C, n_B) \quad = \quad (2^{153.8},\ 6208);\ \ (2^{146},\ 1048576). \tag{7.9}$$

Executing these attacks is more expensive than one would expect from the simple query analysis, which is 2^{128} queries in a quantum black-box model.

Recent researches on quantum algorithms with limited quantum computing resources, focus on Hamiltonian simulation[213], 3-SAT problem solving[304], supervised classification[305], etc.

References

[1] Benioff P, The computer as a physical system, *Journal of Statistical Physics*, 1980, **22**: 563–591.

[2] Feynman R, Simulating physics with computers, *International Journal of Theoretical Physics*, 1982, **21**: 467–488.

[3] Deutsch D, Quantum theory, the Church-Turing principle and the universal quantum computer, *Proc. R. Soc. A*, 1985, **400**: 97–117.

[4] Deutsch D, Quantum computational networks, *Proc. R. Soc. A*, 1989, **425**: 73–90.

[5] Yao A C-C, Quantum circuit complexity, *Proc. FOCS* 1993, IEEE, Palo Alto, 1993, 352–361.

[6] Bernstein E and Vazirani U V, Quantum complexity theory, *SIAM J. Comput.*, 1997, **26**(5): 1411–1473.

[7] Deutsch D and Jozsa R, Rapid solution of problems by quantum computation, *Proc. R. Soc. A*, 1992, **439**: 553–558.

[8] Simon D R, On the power of quantum computation, *SIAM J. Comput.*, 1997, **26**(5): 1474–1483.

[9] Shor P W, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.*, 1997, **26**(5): 1484–1509.

[10] Kitaev A Y, Quantum measurements and the abelian stabilizer problem, arXiv: quant-ph /9511026.

[11] Ettinger M, Høyer P, and Knill E, The quantum query complexity of the hidden subgroup problem is polynomial, *Information Processing Letters*, 2004, **91**(1): 43–48.

[12] Nielsen M A and Chuang I L, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, 2000.

[13] Hallgren S, Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem, *Proc. STOC* 2002, ACM Press, New York, 2002, 653–658.

[14] Proos J and Zalka C, Shor's discrete logarithm quantum algorithm for elliptic curves, *Quantum Information and Computation*, 2003, **3**: 317–344.

[15] Bacon D, Childs A M, and van Dam W, From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups, *Proc. FOCS* 2005, IEEE, Washington, 2005, 469–478.

[16] Chi D P, Kim J S, and Lee S, Notes on the hidden subgroup problem on some semi-direct product groups, *Phys. Lett. A*, 2006, **359**(2): 114–116.

[17] Inui Y and Le Gall G, Efficient quantum algorithms for the hidden subgroup problem over a class of semi-direct product groups, *Quantum Information and Computation*, 2007, **7**(5 & 6): 559–570.

[18] Kuperberg G, A subexponential-time quantum algorithm for the dihedral hidden subgroup problem, *SIAM J. Comput.*, 2005, **35**(1): 170–188.

[19] Moore C, Rockmore D, Russell A, et al., The power of basis selection in Fourier sampling: The hidden subgroup problem in affine groups, *Proc. SODA* 2004, SIAM, Philadelphia, 2004, 1113–1122.

[20] Gavinsky D, Quantum solution to the hidden subgroup problem for poly-near-Hamiltonian-groups, *Quantum Information and Computation*, 2004, **4**: 229–235.

[21] Hallgren S, Russell A, and Ta-Shma A, Normal subgroup reconstruction and quantum computation using group representations, *SIAM J. Comput.*, 2003, **32**(4): 916–934.

[22] Ivanyos G, Magniez F, and Santha M, Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem, *SPAA* 2001, ACM Press, New York, 2001, 263–270.

[23] Grigni M, Schulman L, Vazirani M, et al., Quantum mechanical algorithms for the nonabelian hidden subgroup problem, *Combinatorica*, 2004, **24**: 137–154.

[24] van Dam W, Hallgren S, and Ip L, Quantum algorithms for some hidden shift problems, *SIAM J. Comput.*, 2006, **36**(3): 763–778.

[25] Boneh D and Lipton R J, Algorithms for black-box fields and their application to cryptography, *Advances in Cryptology — CRYPTO*'96, Ed. by Koblitz N, LNCS 1109, 1996, 283–297.

[26] Kuwakado H and Morii M, Quantum distinguisher between the 3-round Feistel cipher and the random permutation, *Proc. ISIT* 2010, IEEE, Austin, TX, 2010, 2682–2685.

[27] Kuwakado H and Morii M, Security on the quantum-type Even-Mansour cipher, *Proc. ISITA* 2012, IEEE, Honolulu, HI, 2012, 312–316.

[28] Santoli T and Schaffner C, Using Simon's algorithm to attack symmetric-key cryptographic primitives, arXiv: 1603.07856 [quant-ph].

[29] Kaplan M, Leurent G, and Leverrier A, Breaking symmetric cryptosystems using quantum period finding, arXiv: 1602.05973v3 [quant-ph].

[30] Ajtai M and Dwork C, A public-key cryptosystem with worst-case/average-case equivalence, *Proc. STOC* 1997, ACM Press, New York, 1997, 284–293.

[31] Regev O, New lattice-based cryptographic constructions, *Journal of the ACM*, 2004, **51**(6): 899–942.

[32] Regev O, Quantum computation and lattice problems, *SIAM J. Comput.*, 2004, **33**(3): 738–760.

[33] Galbraith S and Stolbunov A, Improved algorithm for the isogeny problem for ordinary elliptic

curves, *Applicable Algebra in Engineering, Communication and Computing*, 2013, **24**(2): 107–131.

[34] Couveignes J M, *Hard Homogeneous Spaces*, https://eprint.iacr.org/2006/291.pdf.

[35] Rostovtsev A and Stolbunov A, Public-key cryptosystem based on isogenies, https://eprint.iacr.org/2006/145.pdf.

[36] Stolbunov A, Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves, *Adv. Math. Commun.*, 2010, **4**(2): 215–235.

[37] Childs A M, Jao D, and Soukharev V, Constructing elliptic curve isogenies in quantum subexponential time, *J. Mathematical Cryptology*, 2014, **8**: 1–29.

[38] Grover L K, A fast quantum mechanical algorithm for database search, *Proc. STOC* 1996, ACM Press, New York, 1996, 212–219.

[39] Bennett C H, Bernstein E, Brassard G, et al., Strengths and weaknesses of quantum computing, *SIAM J. Comput.*, 1997, **26**(5): 1510–1523.

[40] Zalka C, Grover's quantum searching algorithm is optimal, *Phy. Rev. A*, 1999, **60**: 2746–2751.

[41] Long G L, Grover algorithm with zero theoretical rate, *Phys. Lett. A*, 2001, **64**: 022307.

[42] Ambainis A, Quantum search algorithms, *SIGACT News*, 2004, **35**(2): 22–35.

[43] Campbell E, Khurana A, and Montanaro A, Applying quantum algorithms to constraint satisfaction problems, arXiv: 1810.05582 [quant-ph].

[44] Liu W Z, Zhang J F, and Long G L, A parallel quantum algorithm for the satisfiability problem, *Common. Theor. Phys.*, 2008, **49**(3): 629–630.

[45] Laarhoven T, Mosca M, and van de Pol J, Solving the shortest vector problem in lattices faster using quantum search, *PQCrypto* 2013, Ed. by Gaborit P, LNCS 7932, Springer, Berlin, Heidelberg, 2013, 83–101, also available: arXiv: 1301.6176v1 [cs.CR].

[46] Faugère J C, Horan K, Kahrobaei D, et al., Fast quantum algorithm for solving multivariate quadratic equations, arXiv: 1712.07211 [cs.CR].

[47] He X Y, Sun X M, Yang G, et al., Exact quantum query complexity of weight decision problems, arXiv: 1801.05717v1 [quant-ph].

[48] Le Gall F and Nishimura H, Quantum algorithms for matrix products over semirings, *Chicago Journal of Theoretical Computer Science*, 2017, **1**: 1–25.

[49] Dürr C and Høyer P, A quantum algorithm for finding the minimum, arXiv: quant-ph/9607014.

[50] Kowada L A B, Lavor C, Portugal R, et al., A new quantum algorithm for solving the minimum searching problem, *International Journal of Quantum Information*, 2008, **6**(3): 427–436.

[51] Brassard G, Høyer P, and Tapp A, Quantum Counting, *Automata, Languages and Programming*, Eds. by Larsen K G, et al., LNCS 1443, Springer, Berlin, Heidelberg, 1998, 820–831.

[52] Brassard G, Høyer P, and Mosca M, Quantum amplitude amplification and estimation, *Quantum Computation and Quantum Information*, 2002, **305**: 53–74.

[53] Brassard G, Høyer P, and Tapp A, Quantum cryptanalysis of hash and claw-free functions, *LATIN'98: Theoretical Informatics*, Eds. by Lucchesi C L and Moura A V, LNCS 1380, Springer, Berlin, Heidelberg, 1998, 163–169.

[54] Aaronson S and Shi Y, Quantum lower bounds for the collision and the element distinctness problems, *Journal of the ACM*, 2004, **51**(4): 595–605.

[55] Wang X, Yao A, and Yao F, *Cryptanalysis on SHA*-1, http://csrc.nist.gov/groups/ST/hash/documents/Wang SHA1-New-Result.pdf.

[56] Cochran M, Notes on the Wang, et al. 2^{63} SHA-1 Differential Path, https://eprint.iacr.

org/2007/474.pdf.

[57] Hoffstein J, Pipher J, and Silverman J H, NTRU: A ring-based public key cryptosystem, *Algorithmic Number Theory*, Ed. by Buhler J P, LNCS 1423, Springer, Berlin, Heidelberg, 1998, 267–288.

[58] Fluhrer S, Quantum cryptanalysis of NTRU, Cryptology ePrint Archive: Report 2015/676, 2015.

[59] Childs A M, Universal computation by quantum walk, *Phys. Rev. Lett.*, 2009, **102**: 180501.

[60] Magniez F, Santha M, and Szegedy M, Quantum algorithms for the triangle problem, *SIAM J. Comput.*, 2007, **37**(2): 413–424.

[61] Jeffery S, Kothari R, and Magniez F, Nested quantum walks with quantum data structures, *Proc. SODA* 2013, SIAM, Philadelphia, 2013, 1474–1485.

[62] Belovs A and Reichardt B W, Span programs and quantum algorithms for st-connectivity and claw detection, *European Symp. on Algorithms*, Eds. by Epstein L, et al., LNCS 7501, Springer, Berlin, Heidelberg, 2012, 193–204.

[63] Buhrman H, Cleve R, de Wolf R, et al., Bounds for small-error and zero-error quantum algorithms, *Proc. FOCS* 1999, IEEE, New York, 1999, 358–368.

[64] Dürr C, Heiligman M, and Høyer P, Quantum query complexity of some graph problems, *SIAM J. Comput.*, 2006, **35**(6): 1310–1328.

[65] Ambainis A, Kempe J, and Rivosh A., Coins make quantum walks faster, *Proc. SODA* 2005, SIAM, Philadelphia, 2005, 1099–1108.

[66] Tulsi A, Faster quantum-walk algorithm for the two-dimensional spatial search, *Phys. Rev. A*, 2008, **78**: 012310.

[67] Magniez F, Nayak A, Richter P C, et al., On the hitting times of quantum versus random walks, *Algorithmica*, 2012, **63**(1): 91–116.

[68] Aaronson S and Ambainis A, Quantum search of spatial regions, *Theory of Computing*, 2005, **1**: 47–79.

[69] Childs A M, Cleve R, Jordan S P, et al., Discrete-query quantum algorithm for NAND trees, *Theory of Computing*, 2009, **5**: 119–123.

[70] Ambainis A, Childs A M, Reichardt B W, et al., Any AND-OR formula of size N can be evaluated in time $n^{1/2+o(1)}$ on a quantum computer, *SIAM J. Comput.*, 2010, **39**(6): 2513–2530.

[71] Reichardt B W, Faster quantum algorithm for evaluating game trees, *Proc. SODA* 2011, SIAM, Philadelphia, 2011, 546–559.

[72] Reichardt B W, Reflections for quantum query algorithms, *Proc. SODA* 2011, SIAM, Philadelphia, 2011, 560–569.

[73] Reichardt B W and Spalek R, Span-program-based quantum algorithm for evaluating formulas, *Theory of Computing*, 2012, **8**: 291–319.

[74] Childs A M and Kothari R, Quantum query complexity of minor-closed graph properties, *SIAM Journal on Computing*, 2012, **41**(6): 1426–1450.

[75] Le Gall F, Improved quantum algorithm for triangle finding via combinatorial arguments, *Proc. FOCS* 2014, IEEE, Philadelphia, 2014, 216–225.

[76] Lee T, Magniez F, and Santha M, Improved quantum query algorithms for triangle finding and associativity testing, *Algorithmica*, 2017, **77**: 459–486.

[77] Bernstein D J, Jeffery S, Lange T, et al., Quantum algorithms for the subset-sum problem, *Post-Quantum Cryptography*, Ed. by Gaborit P, LNCS 7932, Springer, Berlin, Heidelberg, 2013, 16–33.

[78] Ambainis A, Quantum walk algorithm for element distinctness, *SIAM J. Comput.*, 2007, **37**: 210–239.

[79] Magniez F and Nayak A, Quantum complexity of testing group commutativity, *Algorithmica*, 2007, **48**(3): 221–232.

[80] Buhrman H and Spalek R, Quantum verification of matrix products, *Proc. SODA* 2006, SIAM, Philadelphia, 2006, 880–889.

[81] Le Gall F, Improved output-sensitive quantum algorithms for Boolean matrix multiplication, *Proc. SODA* 2012, SIAM, Philadelphia, 2012, 1464–1476.

[82] Dorn S and Thierauf T, The quantum query complexity of algebraic properties, *Fundamentals of Computation Theory*, Eds. by Csuhaj-Varjú E, et al, LNCS 4639, Springer, Berlin, Heidelberg, 2007, 250–260.

[83] Feynman R, Quantum mechanical computer, *Optics News*, 1985, **11**: 11–20.

[84] Chase B A and Landhal A J, Universal quantum walks and adiabatic algorithms by 1d Hamiltonians, arXiv: 0802.1207 [quant-ph].

[85] Farhi E and Gutmann S, Quantum computation and decision trees, *Phys. Rev. A*, 1998, **58**: 915–928.

[86] Meyer D, From quantum cellular automata to quantum lattice gases, *J. Stat. Phys.*, 1996, **85**: 551–574.

[87] Nayak A and Vishvanath A, Quantum walk on the line, arXiv: quant-ph/0010117.

[88] Strauch F W, Connecting the discrete and continuous-time quantum walks, *Phys. Rev. A*, 2006, **74**: 030301.

[89] Childs A M, On the relationship between continuous- and discrete-time quantum walk, *Communications in Mathematical Physics*, 2010, **294**(2): 581–603.

[90] Ambainis A, Bach E, Nayak A, et al., One-dimensional quantum walks, *Proc. STOC* 2001, ACM Press, New York, 2001, 37–49.

[91] Kempe J, Discrete quantum walks hit exponentially faster, *Probability Theory and Related Fields*, 2005, **133**(2): 215–235.

[92] Shenvi N, Kempe J, and Whaley K B, A quantum random-walk search algorithm, *Phys. Rev. A*, 2003, **67**: 052307.

[93] Childs A M, Cleve R, Deotto E, et al., Exponential algorithmic speedup by quantum walk, *Proc. STOC* 2003, ACM Press, New York, 2003, 59–68.

[94] Childs A M, Farhi E, and Gutmann S, An example of the difference between quantum and classical random walks, *Quantum Inf. Process*, 2002, **1**(1 & 2): 35–43.

[95] Szegedy M, Quantum speed-up of markov chain based algorithms, *Proc. FOCS* 2004, IEEE, Rome, 2004, 32–41.

[96] Magniez F, Nayak A, Roland J, et al., Search via quantum walk, *SIAM J. Comput.*, 2011, **40**(1): 142–164.

[97] Childs A M, Jeffery S, Kothari R, et al., A time-efficient quantum walk for 3-distinctness using nested updates, arXiv: 1302.7316 [quant-ph].

[98] Farhi E, Goldstone J, and Gutmann S, A quantum algorithm for the Hamiltonian NAND tree, *Theory of Computing*, 2008, **4**: 169–190.

[99] Harrow A W, Hassidim A, and Lloyd S, Quantum algorithm for solving linear systems of equations, *Phys. Rev. Lett.*, 2009, **103**(15): 150502.

[100] Lloyd S, Universal quantum simulators, *Science*, 1996, **273**(5278): 1073–1078.

[101] Suzuki M, General theory of fractal path integrals with applications to many-body theories and statistical physics, *Journal of Mathematical Physics*, 1991, **32**(2): 400–407.

[102] Aharonov D and Ta-Shma A, Adiabatic quantum state generation and statistical zero knowledge, *Proc. STOC* 2003, ACM Press, New York, 2003, 20–29.

[103] Berry D W, Ahokas G, Cleve R, et al., Efficient quantum algorithms for simulating sparse Hamiltonians, *Comm. Math. Phys.*, 2007, **270**(2): 359–371.

[104] Childs A M and Kothari R, Simulating sparse Hamiltonians with star decompositions, *Theory of Quantum Computation, Communication, and Cryptography*, Eds. by van Dam W, et al., LNCS 6519, Springer-Verlag Berlin Heidelberg, 2011, 94–103.

[105] Berry D W and Childs A M, Black-box Hamiltonian simulation and unitary implementation, *Quantum Information and Computation*, 2012, **12**: 29–62.

[106] Berry D W, Childs A M, Cleve R, et al., Simulating Hamiltonian dynamics with a truncated Taylor series, *Phys. Rev. Lett.*, 2015, **114**: 090502.

[107] Berry D W, Childs A M, and Kothari R, Hamiltonian simulation with nearly optimal dependence on all parameters, *Proc. FOCS* 2015, IEEE, Berkeley, 2015, 792–809.

[108] Low G H and Chuang I L, Hamiltonian simulation by qubitization, arXiv: 1610.06546v2 [quant-ph].

[109] Chakraborty S, Gilyén A, and Jeffery S, The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation, arXiv: 1804.01973v1 [quant-ph].

[110] Gilyén A, Su Y, Low G H, et al., Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics, arXiv: 1806.01838 [quant-ph].

[111] Childs A M and Kothari R, Limitations on the simulation of non-sparse Hamiltonians, *Quantum Information and Computation*, 2010, **10**: 669–684.

[112] Rebentrost P, Steffens A, and Lloyd S, Quantum singular value decomposition of non-sparse low-rank matrices, *Phys. Rev. A*, 2018, **97**: 012327.

[113] Kerenidis I and Prakash A, Quantum recommendation system, *Proc. ITCSC* 2017, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, 2017, **49**: 1–21.

[114] Wang C H and Wossnig L, A quantum algorithm for simulating non-sparse Hamiltonians, arXiv: 1803.08273v1 [quant-ph].

[115] Childs A M, Kothari R, and Somma R D, Quantum algorithm for systems of linear equations with exponentially improved dependence on precision, *SIAM J. Comput.*, 2017, **46**: 1920–1950.

[116] Ambainis A, Variable time amplitude amplification and quantum algorithms for linear algebra problems, *Proc. STACS* 2012, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, 2012, 636–647.

[117] Saad Y, *Iterative Methods for Sparse Linear Systems*, 2nd edition, Society for Industrial and Applied Mathematics, 2003.

[118] Clader B D, Jacobs B C, and Sprouse C R, Preconditioned quantum linear system algorithm, *Phys. Rev. Lett.*, 2013, **110**: 250504.

[119] Wossnig L, Zhao Z K, and Prakash A, Quantum linear system algorithm for dense matrices, *Phys. Rev. Lett.*, 2018, **120**: 050502.

[120] Chen Y A and Gao X S, Quantum algorithms for Boolean equation solving and quantum algebraic attack on cryptosystems, arXiv: 1712.06239v3 [quant-ph].

[121] Chen Y A, Gao X S, and Yuan C M, Quantum algorithms for optimization and polynomial systems solving over finite fields, arXiv: 1802.03856v2 [quant-ph].

[122] Schuld M and Petruccione F, *Supervised Learning with Quantum Computers*, Springer, 2018.

[123] Wittek P, *Quantum Machine Learning: What Quantum Computing Mean to Data Mining*, Academic Press, 2014.

[124] Wiebe N, Braun D, and Lloyd S, Quantum algorithm for data fitting, *Phys. Rev. Lett.*, 2012, **109**(5): 050505.

[125] Schuld M, Sinayskiy I, and Petruccione F, Prediction by linear regression on a quantum computer, *Phys. Rev. A*, 2016, **94**: 022342.

[126] Wang G M, Quantum algorithm for linear regression, *Phy. Rev. A*, 2017, **96**: 012335.

[127] Lloyd S, Mohseni M, and Rebentrost P, Quantum algorithms for supervised and unsupervised machine learning, arXiv: 1307.0411v2 [quant-ph].

[128] Lloyd S, Rebentrost P, and Mohseni M, Quantum principal component analysis, *Nature Physics*, 2014, **10**: 631–633.

[129] Rebentrost P, Mohseni M, and Lloyd S, Quantum support vector machine for big data classification, *Phys. Rev. Lett.*, 2014, **113**(13): 130503.

[130] Rebentrost P, Bromley T R, Weedbrook C, et al., Quantum recurrent neural network, *Phys. Rev. A*, 2018, **98**: 042308.

[131] Altaisky M V, Quantum neural network, arXiv: quant-ph/0107012v2.

[132] Schuld M, Sinayskiy I, and Petruccione F, The quest for a quantum neural network, *Quantum Inf. Process*, 2014, **13**: 2567–2586.

[133] Schuld M, Sinayskiy I, and Petruccione F, Simulating a perceptron on a quantum computer, *Phys. Lett. A*, 2015, **379**: 660–663.

[134] Wan K H, Dahlsten O, Kristjánsson H, et al., Quantum generalisation of feedforward neural networks, *NPJ Quantum Information*, 2017, **3**(1): 36–43.

[135] Wiebe N, Kapoor A, and Svore K, Quantum perceptron models, *Advances in Neural Information Processing Systems*, 2016, **29**: 3999–4007.

[136] Yamamoto A Y, Sundqvist K M, Li P, et al., Simulation of a multidimensional input quantum perceptron, *Quantum Inf. Process*, 2018, **17**(6): 128–139.

[137] Schützhold R, Pattern recognition on a quantum computer, *Phys. Rev. A*, 2003, **67**: 062311.

[138] Trugenberger C A, Quantum pattern recognition, *Quantum Inf. Process*, 2002, **1**(6): 471–493.

[139] Ventura D and Martinez T, Quantum associative memory, *Information Sciences*, 2000, **124**(1): 273–296.

[140] Low G H, Yoder T J, and Chuang I L, Quantum inference on Bayesian networks, *Phys. Rev. A*, 2014, **89**: 062315.

[141] Sentís G, Calsamiglia J, Muñoz-Tapia R, et al., Quantum learning without quantum memory, *Scientific Reports*, 2012, **2**(708): 1–8.

[142] Barry J, Barry D T, and Aaronson S, Quantum POMDPs, *Phys. Rev. A*, 2014, **90**: 032311.

[143] Clark L A, Huang W, Barlow T M, et al., Hidden quantum Markov models and open quantum systems with instantaneous feedback, *Interdisciplinary Symposium on Complex Systems*, Eds. by Sanayei A, et al., ECC 14, Springer, Cham, 2015, 143–151.

[144] Adachi S H and Henderson M P, Application of quantum annealing to training of deep neural networks, arXiv: 1510.06356 [quant-ph].

[145] Wiebe N, Kapoor A, and Svore K, Quantum deep learning, arXiv: 1412.3489 [quant-ph].

[146] Amin M H, Andriyash E, Rolfe J, et al., Quantum Boltzmann machine, *Phys. Rev. X*, 2018, **8**: 021050.

[147] Crawford D, Levit A, Ghadermarzy N, et al., Reinforcement learning using quantum Boltzmann machines, arXiv: 1612.05695 [quant-ph].

[148] Kieferova M and Wiebe N, Tomography and generative data modeling via quantum Boltzmann training, *Phys. Rev. A*, 2017, **96**: 062327.

[149] Messiah A, *Quantum Mechanics*, Vol. II. Wiley, New Jersey, 1976.

[150] Farhi E, Goldstone J, Gutmann S, et al., Quantum computation by adiabatic evolution, arXiv: quant-ph/0001106v1.

[151] Childs A M, Farhi E, and Preskill J, Robustness of adiabatic quantum computation, *Phys. Rev. A*, 2001, **65**: 012322.

[152] Roland J and Cerf N, Quantum search by local adiabatic evolution, *Phys. Rev. A*, 2002, **65**: 042308.

[153] Aharonov D, van Dam W, Kempe J, et al., Adiabatic quantum computation is equivalent to standard quantum computation, *SIAM J. Comput.*, 2007, **37**(1): 166–194.

[154] Childs A M, Farhi E, Goldstone J, et al., Finding cliques by quantum adiabatic evolution, *Quantum Information and Computation*, 2002, **2**(181): 181–191.

[155] Farhi E, Goldstone F, Gutmann S, et al., A quantum adiabatic evolution algorithm applied to instances of an NP-complete problem, *Science*, 2001, **292**(5516): 472–475.

[156] Hogg T, Adiabatic quantum computing for random satisfiability problems, *Phys. Rev. A*, 2003, **67**: 022314.

[157] Reichardt B W, The quantum adiabatic optimization algorithm and local minima, *Proc. STOC* 2004, ACM Press, New York, 2004, 502–510.

[158] van Dam W and Vazirani U V, Limits on quantum adiabatic optimization, http:// citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.394.2905&rep=rep1&type=pdf.

[159] Neven H, Denchev V S, Rose E, et al., Training a binary classifier with the quantum adiabatic algorithm, arXiv: 0811.0416 [quant-ph].

[160] Pudenz K L and Lidar D A, Quantum adiabatic machine learning, *Quantum Inf. Process*, 2013, **12**(5): 2027–2070.

[161] Gaitan F and Clark L, Ramsey numbers and adiabatic quantum computing, *Phys. Rev. Lett.*, 2012, **108**: 010501.

[162] Gaitan F and Clark L, Graph isomorphism and adiabatic quantum computing, *Phys. Rev. A*, 2014, **89**(2): 022342.

[163] Kitaev A Y, Fault-tolerant quantum computation by anyons, *Annals of Physics*, 2003, **303**(1): 2–30.

[164] Freedman M H, Kitaev A, Larsen M J, et al., Topological quantum computation, *Bull. Amer. Math. Soc.*, 2003, **40**: 31–38.

[165] Aharonov D, Jones V, and Landau Z, A polynomial quantum algorithm for approximating the jones polynomial, *Proc. STOC* 2006, ACM Press, New York, 2006, 427–436.

[166] Aharonov D, Arad I, Eban E, et al., Polynomial quantum algorithms for additive approximations of the potts model and other points of the tutte plane, arXiv: quant-ph/0702008.

[167] Wocjan P and Yard J, The Jones polynomial: Quantum algorithms and applications in quantum complexity theory, *Quantum Information and Computation*, 2008, **8**(1): 147–180.

[168] Arad I and Landau Z, Quantum computation and the evaluation of tensor networks, *SIAM J. Comput.*, 2010, **39**(7): 3089–3121.

[169] Aaronson S, The limits of quantum computers, *Scientific American*, 2008, **298**(3): 62–69.

[170] Aaronson S, BQP and the polynomial hierarchy, *Proc. STOC* 2010, ACM Press, New York, 2010, 141–150.

[171] Mahadev U, Classical Verification of Quantum Computations, arXiv: 1804.01082v2 [quant-ph].

[172] Cirac J and Zoller P, Quantum computation with cold trapped ions, *Phys. Rev. Lett.*, 1995, **74**: 4091–4094.

[173] Gershenfeld N and Chuang I L, Bulk spin resonance quantum computing, *Science*, 1997, **275**: 350–356.

[174] Loss D and DiVincenzo D, Quantum computation with quantum dots, *Phys. Rev. A*, 1998, **57**: 120–126.

[175] Vandersypen L M K, Steffen M, Breyta G, et al., Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance, *Nature*, 2001, **414**(6866): 883–887.

[176] Zhao Z, Chen Y A, Zhang A N, et al., Experimental demonstration of five-photon entanglement and open-destination teleportation, *Nature*, 2004, **430**(6995): 54–58.

[177] 12-qubits reached in quantum information quest, https://www.sciencedaily.com/releases/2006/05/060508164700.htm.

[178] World's First 28 qubit Quantum Computer Demonstrated Online at Supercomputing 2007 Conference, https://www.nanowerk.com/news/newsid=3274.php.

[179] Dwave System's 128 qubit chip has been made, https://www.nextbigfuture.com/2008/12/ dwave-systems-128-qubit-chip-has-been.html.

[180] Monz T, Schindler P, Barreiro J T, et al., 14-Qubit Entanglement: Creation and Coherence, *Phys. Rev. Lett.*, 2011, **106**(13): 130506.

[181] D-wave systems breaks the 1000 qubit quantum computing barrier, https://www.dwavesys.com/press-releases/d-wave-systems-breaks-1000-qubit-quantum-computing-barrier.

[182] O'Malley P J J, Babbush R, Kivlichan I D, et al., Scalable quantum simulation of molecular energies, *Phys. Rev. X*, 2016, **6**: 031007.

[183] IBM just made a 17 qubit quantum processor, its most powerful one yet, https://motherboard.vice.com/en_us/article/wnwk5w/ibm-17-qubit-quantum-processor -computer-google.

[184] Quantum inside: Intel manufactures an exotic new chip, https://www.technologyreview.com/s/609094/quantum-inside-intel-manufactures-an-exotic-new-chip/.

[185] IBM Q Experience, https://quantumexperience.ng.bluemix.net/qx/experience.

[186] Coles P J, Eidenbenz S, Pakin S, et al., Quantum Algorithm Implementations for Beginners, arXiv: 1804.03719v1 [cs.ET].

[187] China builds five qubit quantum computer sampling and will scale to 20 qubits by end of this year and could any beat regular computer next year, https://www.nextbigfuture.com/2017/05/china-builds-ten-qubit-quantum-computer-and-will-scale-to-20-qubits-by-end-of -this-year-and-could-any-beat-regular-computer-next-year.html.

[188] IBM raises the bar with a 50-qubit quantum computer, https://www.technologyreview.com/s/609451/ibm-raises-the-bar-with-a-50-qubit-quantum-computer/.

[189] D-wave announces d-wave 2000q quantum computer and first system order, https://www.dwavesys.com/press-releases/d-wave-announces-d-wave-2000q-quantum-computer-and -first-system-order.

[190] CES 2018: Intel's 49-qubit chip shoots for quantum supremacy, https://spectrum.ieee.org/ tech-talk/computing/hardware/intels-49qubit-chip-aims-for-quantum-supremacy.

[191] Google moves toward quantum supremacy with 72-qubit computer, https://www.sciencenews.

org/article/google-moves-toward-quantum-supremacy-72-qubit-computer.

[192] Lu C Y, Browne D E, and Yang T, Demonstration of a compiled version of Shor's quantum factoring algorithm Using Photonic Qubits, *Phys. Rev. Lett.*, 2007, **99**: 250504.

[193] Lanyon B P, Weinhold T J, Langford N K, et al., Experimental demonstration of a compiled version of Shor's algorithm with quantum entanglement, *Phys. Rev. Lett.*, 2007, **99**(25): 250505.

[194] Martin-Lopez E, Laing A, Lawson T, et al., Experimental realisation of Shor's quantum factoring algorithm using qubit recycling, *Nat. Photon.*, 2012, **6**: 773–776.

[195] Chuang I L, Gershenfeld N, and Kubinec M, Experimental implementation of fast quantum searching, *Phys. Rev. Lett.*, 1998, **80**: 3408–3411.

[196] Vandersypen L M K, Steffen M, Sherwood M H, et al., Implementation of a three-quantum-bit search algorithm, *Appl. Phys. Lett.*, 2000, **76**: 646–648.

[197] Barz S, Kassal I, and Ringbauer M, Solving systems of linear equations on a quantum computer, *Sci. Rep.*, 2014, **4**: 115.

[198] Cai X D, Weedbrook C, Su Z E, et al., Experimental quantum computing to solve systems of linear equations, *Phys. Rev. Lett.*, 2013, **110**(23): 230501.

[199] Pan J W, Cao Y D, Yao X W, et al., Experimental realization of quantum algorithm for solving linear systems of equations, *Phys. Rev. A*, 2014, **89**: 022313.

[200] Ambainis A, New Developments in Quantum Algorithms, *Mathematical Foundations of Computer Science*, Eds. by Hlineny P, et al., LNCS 6281, Springer, Berlin, Heidelberg, 2010, 1–11.

[201] Cleve R, Ekert A, Macchiavello C, et al., Quantum algorithms revisited, *Proc. R. Soc. A*, 1997, **454**(1969): 339–354.

[202] Montanaro A, Quantum algorithms: An overview, *npj Quantum Information*, 2016, **2**: 15023.

[203] Shor P W, Progress in quantum algorithms, *Quantum Inf. Process*, 2004, **3**: 5–13.

[204] Sun X M, A survey on quantum computing, *Sci. China Inf. Sci.*, 2016, **8**: 982–1002 (in Chinese).

[205] Dervovic D, Herbster M, Mountney P, et al., Quantum linear systems algorithms: A primer, arXiv: 1802.08227v1 [quant-ph].

[206] Kaye P, Laflamme R, and Mosca M, *An Introduction to Quantum Computing*, Oxford University Press, New York, 2007.

[207] Rieffel E and Polak W, *Quantum Computing — A Gentle Introduction*, The MIT Press, Cambridge, Massachusetts London, England, 2011.

[208] Ambainis A, Quantum walks and their algorithmic applications, *International Journal of Quantum Information*, 2003, **1**: 507–518.

[209] Elías S, Venegas-Andraca S E, Quantum walks: A comprehensive review, *Quantum Inf. Process*, 2012, **11**(5): 1015–1106.

[210] Nayak A, Richter P C, and Szegedy M, Quantum analogs of markov chains, *Encyclopedia of Algorithms*, Ed. by Kao M Y, Springer, Berlin, Heidelberg, 2014, 1–10.

[211] Santha M, Quantum walk based search algorithms, *Proc. TAMC* 2008, Xi'an, 2008, 31–46.

[212] Kempe J, Quantum random walks - an introductory overview, *Contemporary Physics*, 2003, **44**(4): 307–327.

[213] Childs A M, Maslov D, Nam Y, et al., Toward the first quantum simulation with quantum speedup, *Proceedings of the National Academy of Sciences*, 2018, **115**: 9456–9461.

[214] Adcock J C, Allen E, Day M, et al., Advances in quantum machine learning, arXiv: 1512.02900v1 [quant-ph].

[215] Arunachalam S and de Wolf R, *A Survey of Quantum Learning Theory*, arXiv:1701.06806v3

[quant-ph].

[216] Biamonte J, Wittek P, Pancotti N, et al., Quantum machine learning, *Nature*, 2017, **549**: 195–202.

[217] Ciliberto C, Herbster M, Ialongo A D, et al., Quantum machine learning: A classical perspective, *Proc. R. Soc. A*, 2018, **474**(2209): 20170551.

[218] Dunjko V and Briegel H J, Machine learning & artificial intelligence in the quantum domain, arXiv: 1709.02779v1 [quant-ph].

[219] Schuld M, Sinayskiy I, and Petruccione F, An introduction to quantum machine learning, *Contemporary Physics*, 2015, **56**(2): 172–185.

[220] Albash T and Lidar D A, Adiabatic quantum computing, *Rev. Mod. Phys.*, 2018, **90**: 015002.

[221] Quantum algorithm zoo, https://math.nist.gov/quantum/zoo/.

[222] Childs A M, Lecture notes on quantum algorithms, http://www.cs.umd.edu/ amchilds/qa/.

[223] Christopher M D and Nielsen A, The Solovay-Kitaev algorithm, *Quantum Information and Computation*, 2006, **6**(1): 81–95.

[224] Abrams D S and Lloyd S, Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors, *Phys. Rev. Lett.*, 1999, **83**(24): 5162–5165.

[225] Buhrman H, Cleve R, Watrous J, et al., Quantum fingerprinting, *Phys. Rev. Lett.*, 2001, **87**(16): 167902.

[226] Long G L, General quantum interference principle and duality computer, *Common. Theor. Phys.*, 2006, **45**: 825–844.

[227] Childs A M and Wiebe N, Hamiltonian simulation using linear combinations of unitary operations, *Quantum Information and Computation*, 2012, **12**: 901–924.

[228] Kerenidis I and Prakash A, Quantum gradient descent for linear systems and least squares, arXiv: 1704.04992v3 [quant-ph].

[229] Rebentrost P, Schuld M, Wossnig L, et al., Quantum gradient descent and Newton's method for constrained polynomial optimization, arXiv: 1612.01789v2 [quant-ph].

[230] Grover L K and Rudolph T, Creating superpositions that correspond to efficiently integrable probability distributions, arXiv: quant-ph/0208112.

[231] Soklakov A N and Schack R, Efficient state preparation for a register of quantum bits, *Phys. Rev. A*, 2006, **73**: 012307.

[232] Alpaydin E, *Introduction to Machine Learning*, 3rd Edition, The MIT Press, Massachusetts, 2015.

[233] Mackay D, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, Cambridge, 2003.

[234] Sra S, Nowozin S, and Wright S J, *Optimization for Machine Learning*, The MIT Press, Massachusetts, 2011.

[235] Golub G H and Van Loan C F, *Matrix Computations*, 4th Edition, The John Hopkins University Press, Baltimore, MD, 2013.

[236] Hestenes M R and Stiefel E, Methods of conjugate gradients for solving linear systems, *J. Res. Natl. Bur. Stand.*, 1952, **49**: 409–436.

[237] Aaronson S, Quantum Machine Learning Algorithms: Read the Fine Print, *Nature Physics*, 2015, **11**(4): 291–293.

[238] Childs A M, Quantum algorithms: Equation solving by simulation, *Nature Physics*, 2009, **5**(5): 861.

[239] Harrow A W, Quantum algorithms for systems of linear equations, *Encyclopedia of Algorithms*,

Ed. by Kao M Y, Springer, Berlin, Heidelberg, 2016, 1680–1683.

[240] Giovannetti V, Lloyd S, and Maccone L, Architectures for a quantum random access memory, *Phys. Rev. A*, 2008, **78**: 052310.

[241] Giovannetti V, Lloyd S, and Maccone L, Quantum random access memory, *Phys. Rev. Lett.*, 2008, **100**: 160501.

[242] Kerenidis I and Prakash A, A quantum interior point method for LPs and SDPs, arXiv: 1808.09266v1 [quant-ph].

[243] Bardet M, Faugère J C, Salvy B, et al., On the complexity of solving quadratic boolean systems, *Journal of Complexity*, 2013, **29**(1): 53–75.

[244] Hastie T, Tibshirani R, and Friedman J, *The Elements of Statistical Learning*, Springer, New York, 2001.

[245] Haykin S, *Neural Networks and Learning Machines*, 3rd Edition, Pearson, 2009.

[246] Cleveland W S, Robust locally weighted regression and smoothing scatterplots, *Journal of the American Statistical Association*, 1979, **74**(368): 829–836.

[247] Cleveland W S and Devlin S J, Locally-weighted regression: An approach to regression analysis by local fitting, *Journal of the American Statistical Association*, 1988, **83**(403): 596–610.

[248] Ng A, Supervised learning, discriminative algorithms, http://cs229.stanford.edu/notes/ cs229-notes1.pdf.

[249] Suykens J A K and Vandewalle J, Least squares support vector machine classifiers, *Neural Processing Letters*, 1999, **9**(3): 293–300.

[250] Levin D A, Peres Y, and Wilmer E L, *Markov Chains and Mixing Times*, American Mathematical Society, Providence, Rhode Island, 2009.

[251] Lovász L, Random walks on graphs — A survey, *Combinatorics*, 1993, 1–46.

[252] Gerschgorin S, Über die Abgrenzung der Eigenwerte einer Matrix, *Izv. Akad. Nauk. USSR Otd. Fiz.-Mat. Nauk*, 1931, **7**: 749–754.

[253] Høyer P and Komeili M, Efficient quantum walk on the grid with multiple marked elements, *STACS* 2017, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, 2017, 42:1–42:14.

[254] Watrous J, Quantum simulations of classical random walks and undirected graph connectivity, *Journal of Computer and System Sciences*, 2001, **62**(2): 376–391.

[255] Aharonov D, Ambainis A, Kempe J, et al., Quantum walks on graphs, *Proc. STOC* 2001, ACM Press, New York, 2001, 50–59.

[256] Kendon V, Quantum walks on general graphs, *Int. J. Quantum Inf.*, 2006, **4**(5): 791–805.

[257] Potocek V, Gabris A, Kiss T, et al., Optimized quantum random-walk search algorithms on the hypercube, *Phys. Rev. A*, 2009, **79**: 012325.

[258] Tonchev H, Alternative coins for quantum random walk search optimized for a hypercube, *Journal of Quantum Information Science*, 2015, **5**: 6–15.

[259] Ambainis A, Quantum search with variable times, *Theory of Computing Systems*, 2010, **47**(3): 786–807.

[260] Boyer M, Brassard G, Høyer P, et al., Tight bounds on quantum searching, *Fortsch. Phys. Prog. Phys.*, 1998, **46**(4–5): 493–505.

[261] Grover L K, Fixed-point quantum search, *Phys. Rev. Lett.*, 2005, **95**: 150501.

[262] Yoder T J, Low G H, Chuang I L, Optimal fixed-point quantum amplitude amplification using Chebyshev polynomials, *Phys. Rev. Lett.*, 2014, **113**: 210501.

[263] Brassard G, Høyer P, and Tapp A, Quantum algorithm for the collision problem, *ACM SIGACT News*, 1997, **28**: 14–19.

[264] Falk M, Quantum search on the spatial grid, arXiv: 1303.4127 [quant-ph].

[265] Buhrman H, Dürr C, Heiligman M, et al., Quantum algorithms for element distinctness, *SIAM J. Comput.*, 2005, **34**(6): 1324–1330.

[266] Belovs A, Learning-graph-based quantum algorithm for k-distinctness, *Proc. FOCS* 2012, IEEE, New Brunswick, 2012, 207–216.

[267] Jeffery S, Frameworks for quantum algorithms, PhD thesis, University of Waterloo, 2014.

[268] Krovi H, Magniez F, Ozols M, et al., Quantum walks can find a marked element on any graph, *Algorithmica*, 2016, **74**(2): 851–907.

[269] Dohotaru C and Høyer P, Controlled quantum amplification, *Proc. ICALP* 2017, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, 2017, **18**: 1–13.

[270] Ambainis A and Kokainis M, Analysis of the extended hitting time and its properties, *Proc. QIP* 2015, Philadelphia, 2012.

[271] Portugal R, Santos R A M, Fernandes T D, et al., The staggered quantum walk model, *Quantum Information Processing*, 2016, **15**: 85–101.

[272] Aggarwal D, Dadush D, and Regev O, et al., Solving the shortest vector problem in 2^n time via discrete Gaussian sampling, *Proc. STOC* 2014, ACM Press, New York, 2014, 733–742.

[273] Chen Y, Chung K, and Lai C, Space-efficient classical and quantum algorithms for the shortest vector problem, *Quantum Information and Computation*, 2018, **18**(3 & 4): 285–307.

[274] Becker A, Ducas L, and Gama N, et al., New directions in nearest neighbor searching with applications to lattice sieving, *Proc. SODA* 2016, SIAM, Philadelphia, 2016, 10–24.

[275] Laarhoven T, Search problems in cryptography, PhD dissertation, Eindhoven University of Technology, Eindhoven, 2015.

[276] Ettinger M and Høyer P, On quantum algorithms for noncommutative hidden subgroups, *Advances in Applied Mathematics*, 2000, **25**(3): 239–251.

[277] Bacon D, Childs A M, and van Dam W, Optimal measurements for the dihedral hidden subgroup problem, *Chicago Journal of Theoretical Computer Science*, 2006, article 2.

[278] Regev O, A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space, arXiv: quant-ph/0406151.

[279] Babai L, On the complexity of canonical labeling of strongly regular graphs, *SIAM J. Comput.*, 1980, **9**: 212–216.

[280] Spielman D A, Faster isomorphism testing of strongly regular graphs, *Proc. STOC* 1996, ACM Press, New York, 576–584.

[281] Wang F, *The Hidden Subgroup Problem*, Master's Project, Aarhus Universitet, 2010.

[282] Ettinger M and Høyer P, A quantum observable for the graph isomorphism problem, arXiv: quant-ph/9901029.

[283] Beals R, Quantum computation of Fourier transforms over symmetric groups, *Proc. STOC* 1997, ACM Press, New York, 1997, 48–53.

[284] Boneh D and Lipton R J, Quantum cryptanalysis of hidden linear functions, *Advances in Cryptology CRYPTO*'95, Ed. by Coppersmith D, LNCS 963, Springer-Verlag Berlin, Heidelberg, 1995, 424–437.

[285] Hallgren S, Moore C, Roetteler M, et al., Limitations of quantum coset states for graph isomorphism, *J. ACM*, 2010, **57**(6): 1–33.

[286] Moore C, Russell A, and Schulman L J, The symmetric group defies strong fourier sampling, *SIAM J. Comput.*, 2008, **37**(6): 1842–1864.

[287] Kawano Y and Sekigawa H, Quantum fourier transform over symmetric groups, *Proc. ISSAC* 2013, ACM Press, New York, 2013, 227–234.

[288] Kawano Y and Sekigawa H, Quantum Fourier transform over symmetric groups - improved result, *J. Symb. Comp.*, 2016, **75**: 219–243.

[289] Williams V V and Williams R, Subcubic equivalences between path, matrix and triangle problems, *Proc. FOCS* 2010, IEEE, Las Vegas, 2010, 645–654.

[290] Williams V V and Williams R, Finding, minimizing, and counting weighted subgraphs, *SIAM J. Comput.*, 2013, **42**(3): 831–854.

[291] Williams R, A new algorithm for optimal 2-constraint satisfaction and its implications, *Theor. Comput. Sci.*, **348**(2–3): 357–365.

[292] Belovs A, Span programs for functions with constant-sized 1-certificates, *Proc. STOC* 2012, ACM Press, New York, 2012, 77–84.

[293] Grötschel M, Lovász L, and Schrijver A, *Geometric Algorithms and Combinatorial Optimization*, Springer, New York, 1988.

[294] Lee Y T, Sidford A, and Wong S C, A faster cutting plane method and its implications for combinatorial and convex optimization, *Proc. FOCS* 2015, IEEE, Berkeley, 2015, 1049–1065.

[295] Brandão F G S L and Svore K M, Quantum speed-ups for solving semidefinite programs, *Proc. FOCS* 2017, IEEE, Berkeley, 2017, 415–426.

[296] Apeldoorn J van, Gilyén A, Gribling S, et al., Quantum SDP-solvers: Better upper and lower bounds, *Proc. FOCS* 2017, IEEE, Berkeley, 2017, 403–414.

[297] Apeldoorn J van and Gilyén A, Improvements in quantum SDP-solving with applications, arXiv: 1804.05058 [quant-ph].

[298] Brandão F G S L, Kalev A, Li T Y, et al., Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning, arXiv: 1710.02581v2 [quant-ph].

[299] Lee Y T, Sidford A, and Vempala S S, Efficient convex optimization with membership oracles, arXiv: 1706.07357 [cs.DS].

[300] Chakraborty S, Childs A M, Li T Y, et al., Quantum algorithms and lower bounds for convex optimization, arXiv: 1809.01731 [quant-ph].

[301] Apeldoorn J van, Gilyén A, Gribling S, et al., Convex optimization using quantum oracles, arXiv: 1809.00643v1 [quant-ph].

[302] Grassl M, Langenberg B, and Roetteler M, et al., Applying Grover's Algorithm to AES: Quantum Resource Estimates, *PQCrypto* 2016, Ed. by Takagi T, LNCS 9606, Springer, Cham 2016, 29–43.

[303] Amy M, Di Matteo O, and Gheorghiu V, et al., Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3, *Selected Areas in Cryptography*, Eds. by Avanzi R and Heys H, LNCS 10532, Springer, Cham, 2017, 317–337.

[304] Dunjko V, Ge Y M, and Cirac I, Computational speedups using small quantum devices, arXiv: 1807.08970 [quant-ph].

[305] Schuld M, Bocharov A, Svore K, et al., Circuit-centric quantum classifiers, arXiv: 1804.00633 [quant-ph].