17

Loading classical data

The end-to-end quantum applications covered in this book have classical inputs and classical outputs, in the sense that the problem is specified by some set of classical data, and the solution to the problem should be a different set of classical data. In some cases, the input data is relatively small, and loading it into the algorithm does not contribute significantly to the cost of the algorithm. In other cases—for example, "big data" problems within the areas of machine learning and finance—the dominant costs, both for classical and quantum algorithms, can be related to how the algorithms load and manipulate this large quantity of input data. Consequently, the availability of quantum speedups for these problems is often dependent on the ability to quickly and coherently access this data. The true cost of this access is the source of significant subtlety in many end-to-end quantum algorithms.

The authors are grateful to Thomas Häner, Damian Steiger, and Xiao Yuan for reviewing this chapter.

17.1 Quantum random access memory

Rough overview (in words)

Quantum random access memory (QRAM) is a construction that enables coherent access to a database, such that multiple different elements can be read in superposition. The ability to rapidly access large, unstructured datasets in this way is crucial to the speedups of certain quantum algorithms, for example in quantum machine learning based on quantum linear algebra. QRAM is commonly invoked to circumvent data-input bottlenecks [1] in situations where loading input data could dominate the end-to-end runtime of an algorithm. It

remains an open question, however, whether a large-scale dedicated QRAM device will ever be practical, casting doubt on quantum speedups that rely on QRAM. Note that, while here we focus on the more common use case of loading *classical* data with QRAM, certain QRAM architectures can be adapted to also load *quantum* data [427].

Rough overview (in math)

Consider a length-N, unstructured classical data vector x, and denote the i-th entry as x_i. Let the number of bits of x_i be denoted by d. Given an input quantum state $|\psi\rangle = \sum_{i=0}^{N-1} \sum_{b\in\{0,1\}^d} \alpha_{ib}|i\rangle|b\rangle$, QRAM is defined [434] as a unitary operation Q with the action

$$Q|\psi\rangle = Q \sum_{i=0}^{N-1} \sum_{b\in\{0,1\}^d} \alpha_{ib}|i\rangle|b\rangle = \sum_{i=0}^{N-1} \sum_{b\in\{0,1\}^d} \alpha_{ib}|i\rangle|b \oplus x_i\rangle. \tag{17.1}$$

Here, the first $\log_2(N)$-qubit register stores the "address" (assuming for simplicity that N is a power of 2), while the second d-qubit register stores the corresponding "data." Note that the unitary Q can also be understood as an oracle (or black box) providing access to x, as $Q(\sum_i \alpha_i|i\rangle|0\rangle) = \sum_i \alpha_i|i\rangle|x_i\rangle$.

Let T_Q denote the "cost" of implementing the operation Q, where T_Q can be measured in wall time, circuit depth, total number of gates, total circuit spacetime occupancy, total number of T gates, etc., depending on the context. Algorithms that rely on QRAM to claim exponential speedups over their classical counterparts frequently assume that $T_Q = \text{polylog}(N)$. However, as discussed in §Caveats, below, it is crucial to emphasize that this assumption can only hold when T_Q is interpreted as the circuit depth or wall time (or something similar) to implement Q; whereas, if T_Q is taken to be the total gate cost or the spacetime occupied by the computation, simple gate counting arguments imply a lower bound of $T_Q \geq \Omega(dN)$. In a discrete gate set, each unit of spacetime can be occupied with only a finite number of unique gates; since there are 2^{dN} different possible data vectors x, the circuit must have at least $\Omega(dN)$ spacetime to be able to implement all possibilities (see also [568, Section V] for a more detailed discussion).

Dominant resource cost (gates/qubits)

Let us consider for simplicity the $d = 1$ case, that is, when each data entry is a single bit. The QRAM operation Q can be implemented as a quantum circuit that uses $O(N)$ gates. Assuming gates acting on disjoint qubits can be parallelized, a circuit depth of $T_Q = O(\log(N))$ can be achieved at the expense of using $O(N)$ ancillary qubits; explicit circuits can be found in, for example, [349, 496]. The number of ancillary qubits can be traded off for increased

circuit depth; circuits implementing Q can be constructed using $O(N/M)$ ancillary qubits and depth $O(M \log(N))$, where $M \in [1, N]$; see examples in [722, 140, 349, 496] (the setting of $M = N/\log(N)$ is sometimes referred to as "QROM"—see terminology caveats below—and its fault-tolerant cost of implementation is well established [75]).

If the data vector x is sparse—that is, only s of its N entries are nonzero—then there exist circuit implementations with depth as shallow as $T_Q = O(\log(s \log(N)))$, using $O(s \log(s) \log(N))$ ancillary qubits, both of which for $s = O(1)$ are exponentially better than the general case of a dense vector x [1085].

Each of the above constructions can be generalized to the $d > 1$ case with various space-time tradeoffs. For example, the d bits of a data entry can be queried in series, requiring $O(N)$ ancillary qubits with depth $T_Q = O(d \log(N))$ (improvement to $T_Q = O(d + \log(N))$ is possible for certain QRAM architectures [269]). Alternatively, the d bits can be accessed in parallel, with depth $T_Q = O(\log(N))$, but at the price of $O(Nd)$ ancillary qubits.

Caveats

The main concern for QRAM's practicality is the large hardware overhead that is necessary to realize fast queries with depth $T_Q = O(\log(N))$. This cost is likely to be prohibitive for big-data applications where N can be millions or billions. The cost will also be magnified by additional overhead associated with error correction and fault tolerance [349], especially considering that circuits implementing Q are composed of $O(N)$ non-Clifford gates. Indeed, this observation together with the assumption that magic state distillation is expensive to run in a massively parallel fashion, has led some to argue that $T_Q = O(\log(N))$ is not realistic in a fault-tolerant setting (see, e.g., [568]). However, it is possible that alternative approaches to fault tolerance tailored to QRAM could help alleviate this large hardware overhead.

The fault-tolerance overhead may be reduced for the so-called bucket-brigade QRAM (BBQRAM) [434, 62, 496], which is a family of circuits implementing Q that are intrinsically resilient to noise. More precisely, [496] shows that if ϵ is the per-gate error rate, BBQRAM circuits can implement Q with leading-order fidelity $F \sim 1 - \epsilon\,\text{polylog}(N)$, while generic circuits implementing Q have leading-order fidelity $F \sim 1 - \epsilon O(N)$. Nevertheless, at the scale necessary for useful end-to-end applications, some amount of error correction will almost certainly be required even for BBQRAM circuits.

Even if depth $T_Q = \text{polylog}(N)$ is practically achievable, some have argued that any fair comparison with state-of-the-art classical methods should then allow for classical parallel computation. After all, the parallel classical

hardware necessary to operate the circuit Q (including the quantum error correction) could in principle be repurposed directly toward solving the end-to-end computational problem. For example, many linear algebra tasks such as matrix-matrix and matrix-vector multiplication of size-N objects are amenable to parallelization, and can also have cost scaling as polylog(N) in some parallel classical models of computation [954, 568]. Under such a comparison, it becomes difficult to identify conditions where QRAM-based quantum algorithms can give rise to a significant scaling advantage [568].

Some terminology caveats:

- The unitary Q in Eq. (17.1) is referred to by some as quantum read-only memory (QROM) [75], reflecting the fact that Q corresponds only to reading data. Some algorithms also require the ability to write to the vector x during computation, but the writing of classical (i.e., not in superposition) data need not be implemented via a quantum circuit.
- The term QRAM is used by different authors to refer to the unitary Q, families of circuits that implement Q, or quantum hardware that runs said circuits.
- The terms QRAM and QROM are sometimes used for distinguishing the cases of $T_Q = \text{polylog}(N)$ and $T_Q = \text{poly}(N)$, respectively, even though T_Q is unrelated to the distinction between reading and writing. The term QROAM has also been used to describe intermediate circuits that trade off depth and width [140].
- Some use the term QRAM to refer exclusively to the case $N \gg 1$ and $T_Q = \text{polylog}(N)$ depth, where the implementation challenges for QRAM are most pronounced.

Elsewhere in this book, we follow the convention described in the final bullet point above: usage of the term QRAM, unless specified otherwise, refers to the ability to implement Q at cost polylog(N).

Example use cases

- Quantum linear algebra: QRAM can be used as an oracle for implementing linear algebra algorithms operating on unstructured data (e.g., by acting as a subroutine in a block-encoding), with applications in machine learning, finance, etc. For example, the quantum recommendation systems algorithm [608] (now dequantized [976]) uses QRAM as a subroutine to efficiently encode rows of an input data matrix in the amplitudes of quantum states (see Appendix A of [608] for details).
- Hamiltonian simulation, quantum chemistry, condensed matter physics: In the linear combination of unitaries input model, QRAM can be used as a

subroutine for "PREPARE" oracles that encode coefficients of the simulated Hamiltonian into the amplitudes of quantum states [75]. These use cases typically consider the hybrid QROM/QRAM constructions with $O(K \log(N))$ ancillary qubits and depth $O(N/K)$ (with the parameter K to be optimized), because the amount of data (and thus the size of N) scales only polynomially with the system size.

- Grover search: QRAM can be used as an implementation for Grover's oracle in the context of an unstructured database search; see Chapter 4 of [801]. This appears for example in quantum algorithms that utilize dynamic programming to give polynomial speedups for combinatorial optimization problems like the traveling salesperson problem [28]. However, it has been argued that a quantum computer running Grover's algorithm with a QRAM-based oracle would not provide a speedup over a classical computer with comparable hardware resources [954].
- Topological data analysis (TDA): A small QRAM (i.e., not exponentially larger than the main quantum data register) is used in some quantum algorithms for TDA [709, 755] in order to load the positions of the data points for computing whether simplices are present in the complex at a given length scale.

Further reading

- Reference [568] focuses on various fundamental and practical concerns for large-scale QRAM, while also providing a comprehensive survey.
- Reference [293] provides an overview of practical concerns facing QRAM in the context of big-data applications (though the discussions of noise resilience there and in [62] are somewhat outdated, cf. [496]).

17.2 Preparing quantum states from classical data

Rough overview (in words)

An important subroutine in many quantum algorithms is preparing a quantum state given a list of its amplitudes stored, for example, in a classical database.[1] The upshot is that N amplitudes, which require $O(N)$ classical bits to write down, can be encoded in a quantum state with only $\log_2(N)$ qubits, an exponential compression in memory. However, there are caveats; for example, simple information-theoretic bounds [835] dictate that the quantum circuit that

[1] When the amplitudes are given by some well-behaved function, rather than being arbitrarily chosen, different (related) protocols are used; see §Further reading, below.

prepares the $\log_2(N)$-qubit state must still have at least $O(N)$ gates, so no exponential advantage in gate complexity is possible. Additionally, reading out the N amplitudes from the encoded quantum state generally requires full quantum state tomography, requiring $\Omega(N)$ preparations of the state. Depending on which resource is being optimized, the best protocol for state preparation will look different, and optimal state preparation methods are known for several natural choices of metric.

Rough overview (in math)

Let $x = (x_0, \ldots, x_{N-1}) \in \mathbb{C}^N$ be a vector of N complex numbers, where N is a power of 2, and let

$$|\psi\rangle = \frac{1}{||x||} \sum_{i=0}^{N-1} x_i |i\rangle \tag{17.2}$$

be the associated normalized quantum state, where $||x||$ denotes the standard Euclidean vector norm. Let $n = \log_2(N)$ denote the number of qubits of $|\psi\rangle$. The goal is to prepare the state $|\psi\rangle$ by applying a quantum circuit to the initial state $|0\rangle^{\otimes n}$. This problem has been extensively studied in the literature; a common approach, originating in [463], is to iterate through each of the n qubits and perform a single-qubit rotation, with the angle of rotation depending on the setting of the previous qubits. The rotation on the first qubit creates the 1-qubit state

$$\left(\sqrt{\sum_{i=0}^{N/2-1} |x_i|^2} \right) |0\rangle + \left(\sqrt{\sum_{i=N/2}^{N-1} |x_i|^2} \right) |1\rangle$$

by performing a single-qubit rotation (about the Y axis) on the state $|0\rangle$ by an appropriate angle. Next, a similar kind of single-qubit rotation is performed on the second qubit, where the angle of rotation depends on whether the first qubit is $|0\rangle$ or $|1\rangle$. The $(m+1)$st rotation is by one of 2^m angles, depending on the setting of the first m qubits. Thus, in total there are $1 + 2 + \cdots + 2^{n-1} = N - 1$ total angles that might be used for single-qubit rotations. This sequence of operations prepares the state $||x||^{-1} \sum_{i=0}^{N-1} |x_i| |i\rangle$. To apply the phases, a single- (or zero-) qubit phase gate with the appropriate phase "angle" is performed—the angle depends on the setting of all n qubits, corresponding to the $N = 2^n$ different phases that might be needed. Thus, the total number of angles that define the protocol is $2N - 1$, exactly corresponding to the number of real parameters needed to describe the general state in Eq. (17.2).

It remains to describe how the controlled single-qubit rotations are performed when there are many control bits and different angles for each setting of the control. Here, one has many choices and the exact method will depend

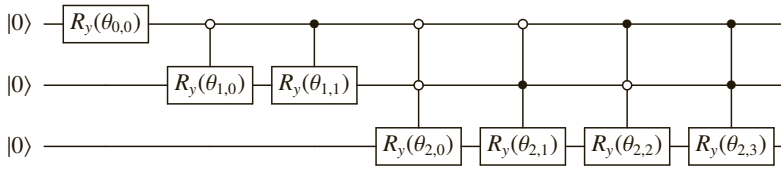

Figure 17.1 A simple quantum circuit to prepare an arbitrary state $|\psi\rangle$ with non-negative real amplitudes on $n = 3$ qubits. The gate $R_y(\theta)$ denotes a single-qubit rotation by angle θ about the Y axis. The $2^n - 1$ angles $\theta_{s,p}$ indexed by $s \in \{0, 1, \ldots, n-1\}$ and $p \in \{0, 1, \ldots, 2^s - 1\}$ can be calculated from the amplitudes x_i. To account for negative or complex amplitudes, 2^n additional controlled phase gates would be needed. More sophisticated proposals can reduce the depth for ancilla-free constructions from $O(2^n)$ to $O(2^n/n)$ [968].

on how one has access to the data in x and what resource is being optimized. The most straightforward way is to iterate through each possible setting of the control bits and perform a multiply controlled rotation by a fixed angle for each in sequence. This approach requires $O(N)$ n-qubit gates applied sequentially, as depicted in Fig. 17.1. Assuming one can perform arbitrary single-qubit gates to exact precision, it is possible to prepare the state $|\psi\rangle$ exactly. However, it is often useful to work with finite precision angles stored in binary, moreover one often needs to design circuits from a discrete gate set, such as the Clifford + T gate set, when compiling into a gate sequence that can be implemented fault tolerantly. When this is the case, single-qubit rotations must be performed approximately: to approximate a single-qubit rotation to error ϵ, a Clifford + T sequence of length $O(\log(1/\epsilon))$ can be applied [884].

When ancilla qubits are available, one can design protocols that have shallower depth (but about the same total number of gates). For example, one can store the $2N - 1$ angles in a quantum random access memory (QRAM) data structure. In this case, for $m = 1, \ldots, n$, to perform the required controlled rotations on the mth qubit in superposition, one can (i) read in the (approximate) binary value of the rotation angle (depending on the setting of the first $m - 1$ qubits) into an ancilla register using a single query to the QRAM, and then (ii) perform fixed-angle single-qubit rotations on the mth qubit, controlled by the qubits of the ancilla register storing the binary representation of the rotation angle, and finally (iii) reset (uncompute) the ancilla register with another query to the QRAM. This way, one applies the correct angle in parallel, rather than iterating through all possible 2^{m-1} angles.

Dominant resource cost (gates/qubits)

In Table 17.1, we collect several state preparation results in the model where any single-qubit gate can be performed exactly and the only multiqubit gates allowed are CNOTs. The circuit size (i.e., the total number of single-qubit and CNOT gates) and depth (i.e., the number of parallel-acting layers of gates), as well as the number of ancilla qubits (i.e., the number of qubits beyond the n qubits needed to hold the state $|\psi\rangle$) are listed.

Ref.	Circuit size	Circuit depth	Ancilla qubits
[968, 1073]	$O(2^n)$	$O(2^n/n)$	none
[968, 1073]	$O(2^n)$	$O(2^n/(m+n))$	$m \in [0, O(2^n/n)]$
[968, 1085, 468]	$O(2^n)$	$O(n)$	$O(2^n)$

Table 17.1 Asymptotic resource cost (and tradeoffs therein) of exact state preparation of abitrary states in a gate set with CNOT gates and arbitrary single-qubit gates.

Note that the result of [1073], which shows depth $O(2^n/(m+n))$ using m ancilla qubits for $m \leq O(2^n/n)$, encompasses all other results in the table (and is superior to the third row as it uses $O(2^n/n)$ ancilla qubits instead of $O(2^n)$). We include the other results for completeness, as they are distinct constructions and can have other potential upsides.

A lower bound of $\Omega(2^n)$ is known for circuit size [835], so all of the results above are size optimal up to constant factors. Moreover, for any m ancillas, a lower bound of $\Omega(\max(n, 2^n/(n+m)))$ is known for the circuit depth [968], so all of the results above are also optimal in circuit depth, up to constant factors.

For approximate state preparation using a discrete gate set such as Clifford + T, the state $|\psi\rangle$ is prepared up to ϵ error in ℓ_2-norm, so that the circuit size and depth depends on ϵ. Results in this model are collected in Table 17.2.

Ref.	Circuit size	Circuit depth	Ancilla qubits
[968]	$O(2^n \log(2^n/\epsilon))$	$O\left(\frac{2^n}{n} \log(2^n/\epsilon)\right)$	none
[968]	$O(2^n \log(2^n/\epsilon))$	$O\left(\frac{2^n}{m+n} \log(2^n/\epsilon)\right)$	$m \in [0, O\left(\frac{2^n}{n \log(n)}\right)]$
[1084]	$O(2^n \log(1/\epsilon))$	$O\left(\frac{2^n}{m} \log(m) \log(\log(m)/\epsilon)\right)$	$m \in [0, O(2^n)]$
[468]	$O(2^n \log(n/\epsilon))$	$O(n + \log(1/\epsilon))$	$O(2^n)$

Table 17.2 Asymptotic resource cost (and tradeoffs therein) of approximate state preparation using the Clifford + T gate set.

If the state $|\psi\rangle$ is sparse, meaning that only s of the N amplitudes are nonzero, then more efficient state preparation methods are known. In particular, [1085, 968, 727] have studied shallow-depth circuits for sparse state

preparation, achieving circuit depth $O(\log(ns))$ using only $O(ns/\log(s))$ ancilla qubits [727], a great improvement over the general case when $s \ll N$.

In some fault-tolerant implementation schemes, such as lattice surgery using surface codes, Clifford gates can be performed cheaply, while T gates require the expensive process of magic state distillation. While $\Omega(2^n \log(1/\epsilon)/\log(n))$ total gates are necessary [801, Eq. 4.85] to approximately create $|\psi\rangle$, [722] noted that it is possible to reduce the number of T gates to $\sqrt{2^n}\log(2^n/\epsilon)$ using $\sqrt{2^n}\log(1/\epsilon)$ ancillas (in fact, there is a smooth tradeoff between the T count and the number of ancillas). Furthermore, these ancillas can be *dirty*, meaning that they can be initialized into any quantum state and they are returned to their (potentially unknown) initial state at the end of the procedure.

All of the above constructions are "garbage-free" state preparation protocols, because they prepare the state $|\psi\rangle$ exactly and all ancilla qubits are returned to their initial state. However, in some applications, it is allowed to leave an ancilla register entangled with the data as long as the amplitudes are correct. That is, one might prepare the state

$$\frac{1}{||x||}\sum_{i=0}^{N-1} x_i|i\rangle \otimes |\text{garbage}_i\rangle\,.$$

In this setting, en route to giving better algorithms for the electronic structure problem, [75, Section IIID] gave a construction that approximately prepares the above state using only $O(N + \log(1/\epsilon))$ T gates, albeit still requiring $O(N\log(1/\epsilon))$ Clifford gates and $O(\log(N/\epsilon))$ ancillas. In [75], the construction is presented with $O(N)$ depth, but it could be improved to $O(\log(N))$ depth at the expense of additional ancillas, using log-depth constructions for QRAM, and it could also be combined with the space-time tradeoffs mentioned above, as discussed in [722, 140].

Caveats

- Classical preprocessing: Computing the circuits for preparing $|\psi\rangle$ given the list of N coefficients x can be a non-negligible classical cost. For example, computing each of the $O(N)$ single-qubit rotation angles requires computing sums and evaluating trigonometric functions, which can be done to precision ϵ in $\text{polylog}(1/\epsilon)$ classical time. Moreover, computing Clifford + T gate sequences that approximate given rotation angles to error ϵ likewise requires $\text{polylog}(1/\epsilon)$ classical time [884]. The total classical work scales as $O(N\,\text{polylog}(1/\epsilon))$, although this cost can be parallelized.
- Coherent arithmetic: To avoid some of the classical preprocessing, one might try to perform the arithmetic coherently. This might be unavoidable if the entries of x arrive in an online fashion and rotation angles and other

quantities need to be computed after superpositions have been created. Formally, the scaling of coherent arithmetic is mild, generally requiring just polylog$(N, 1/\epsilon)$ number of gates and ancilla qubits, but in practice this is likely to be expensive. For example, known methods for coherently computing arcsin$(\cdot)$ to nine bits of precision use order-10^4 Toffoli gates and more than 100 ancilla qubits [494]. See [894] for a general black-box approach that avoids coherent arithmetic.

- Too many ancilla qubits: Achieving depths that scale logarithmically with N requires $O(N)$ ancilla qubits, which limits the size of N that might be practical. This could be mitigated if it is possible to develop a large-scale hardware element specialized for performing the sort of circuits that arise in these protocols, similar to a QRAM.
- Long-range gates: Achieving polylog(N) depth for state preparation requires $O(N)$ ancilla qubits and $O(N)$ gates, many of which act in parallel and on far-separated qubits. If spatial locality were imposed, it would likely be difficult to avoid $O(N)$ overhead in depth.
- Dequantization: Consider the task of drawing samples from the same probability distribution induced by measuring $|\psi\rangle$ in the computational basis in time polylog(N) time. Preparing $|\psi\rangle$ as described is a quantum method of doing so, but the same can be done classically by first constructing a certain classical data structure and assuming access to classical RAM [248]. In some machine learning applications, this idea leads to classical algorithms that effectively dequantize quantum algorithms that utilize the state preparation primitive [976, 977].

Example use cases

- Hamiltonian simulation via linear combination of unitaries (LCU) requires a PREPARE step where a state is prepared with certain classically computed coefficients. Relatedly, the same PREPARE gadget is used to construct block-encodings of such Hamiltonians. However, in this application, state preparation with garbage is generally allowable.
- In certain quantum machine learning protocols, classical data (e.g., image pixel values) are encoded into a quantum state via the so-called "amplitude encoding," where N classical features are stored in a quantum state of $\log_2(N)$ qubits [916]. Following the preparation of the amplitude-encoded data, the state is processed with the goal of, for example, classifying the image.
- Creating a block-encoding of a matrix of classical data is performed using state preparation as a subroutine (more precisely, block-encoding classical data requires controlled state preparation). The block-encoding is then use-

ful in a variety of contexts, for example in quantum interior point methods, and certain quantum machine learning algorithms.

Further reading

- When the amplitudes x_i correspond to an efficiently computable function $f(i)$, the complexity of state preparation can be reduced. In this case, the oracle access to x_i can be replaced by a reversible computation of $f(i)$, up to t bits of precision, using coherent arithmetic $|i\rangle|0^t\rangle \to |i\rangle|f(i)\rangle$ [494, 149, 791]. The value of $f(i)$ can be *transduced* into the amplitude using the methods of [465, 894, 1020, 106], and the success probability boosted to unity using quantum amplitude amplification. There is an alternative method [754], based on quantum singular value transformation (QSVT) that circumvents the need for the coherent evaluation of $f(i)$ by implementing a low-cost block-encoding of $\sin(i)$, and then using QSVT to apply $f(\arcsin(\cdot))$ to this block-encoding. The complexity of both of these approaches depends on an "ℓ_2-norm filling-fraction" $\mathcal{F}_f^{[N]} := \|f(i)\|_2/(\sqrt{N}|f(i)|_{\max})$ as $O(1/\mathcal{F}_f^{[N]})$ (see [754] for more detail). There is also an approach [859] based on the adiabatic algorithm which has a worse dependence on $\mathcal{F}_f^{[N]}$. For efficiently integrable probability distributions, one can use the approach of [463], which has complexity independent of $\mathcal{F}_f^{[N]}$. However, this approach requires coherent arithmetic to reversibly evaluate the integral of the desired function (when applied to functions for which an analytic expression for the integral is not available, this can nullify the quadratic speedup in quantum-accelerated Monte Carlo estimation [521]). There also exist methods specialized for certain target states, such as Gaussians [623, 860].
- A related problem asks to synthesize an arbitrary $2^n \times 2^n$ unitary. Without ancillas, this requires depth and size $O(4^n)$, for which there are upper [787] and lower [931] bounds that match up to constant factors. With ancillas, it is an open question whether or not the depth can be reduced to poly(n); this is related to the "unitary synthesis problem" from the list of open problems in [3], and it has been studied in several works, for example, [968, 883, 1073]. A depth lower bound of $\Omega(n + 4^n/(m + n))$ is known for m ancilla qubits [968], but the shallowest upper bound is depth $O(n2^{n/2})$, using $m = O(4^n/n)$ ancilla qubits [1073].

17.3 Block-encoding dense matrices of classical data

Rough overview (in words)

Many potential applications of quantum algorithms, especially in the area of machine learning, require access to large amounts of classical data, and in order to process this data on quantum devices, one needs coherent query access to the data. Block-encoding is a technique for importing classical data into quantum computers that provides exactly this type of coherent query access. Block-encodings work by encoding the matrices of classical data as blocks within larger matrices, which are defined such that the full encoding is a unitary operator. One way of thinking of this process is by "brute-force" compiling a unitary with the right structure, and then postselecting measurement outcomes to ensure the desired block of the unitary was applied. In general, block-encoding a dense matrix is not an efficient process, as both the normalization factor of the block-encoding and the circuit complexity to implement it can scale with the size of the matrix (e.g., poly(N) for an $N \times N$ matrix). Nonetheless, end-to-end applications often assign *polylogarithmic* cost to the block-encoding, which is achievable if the relevant cost metric is the circuit depth (rather than the circuit size)—this is similar to the assumption that one has access to large-scale log-depth quantum random access memory (QRAM). For a general treatment not restricted to dense classical data, see Section 10.1 on block-encoding.

Rough overview (in math)

Given an $N \times N$ matrix A, a block-encoding is a way of encoding the matrix A as a block in a larger unitary matrix:

$$U_A = \begin{pmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{pmatrix}.$$

If A is not square, one can pad it with zeros such that it becomes square. Let $n = \lceil \log_2(N) \rceil$. We say that the $(n + a)$-qubit unitary U_A is an (α, a, ϵ)-block-encoding of the matrix $A \in \mathbb{C}^{N \times N}$ if

$$\left\| A - \alpha(\langle 0|^{\otimes a} \otimes I) U_A (|0\rangle^{\otimes a} \otimes I) \right\| \leq \epsilon,$$

where $a \in \mathbb{N}$ represents the number of ancilla qubits needed, $\alpha \in \mathbb{R}_+$ is a normalization constant, and $\epsilon \in \mathbb{R}_+$ is an error parameter. The fact that U_A is unitary implies that the normalization constant α must satisfy $\alpha \geq \|A\|$, where $\|\cdot\|$ denotes the spectral norm.

In this section, we consider the case where the N^2 entries of A are arbitrary values provided to us in a classical database. In general, all of these entries

can be nonzero; that is, A is a dense matrix. The goal is to provide a quantum circuit implementing a unitary U_A as above while minimizing quantities such as the circuit depth and circuit size, as well as the number of ancilla qubits a and the normalization constant α. The block-encoding construction we focus on performs U_A using a pair of state preparation unitaries [429, 608, 248], following the more general method of block-encoding Gram matrices [429, Lemma 47]. In particular, the product

$$U_A = U_R^\dagger U_L$$

is an exact $(\alpha, a, 0)$-block-encoding of A, where U_L and U_R are unitaries that perform (controlled) state preparation. Specifically, the $(n + a)$-qubit unitaries U_L and U_R prepare a different $2n$-qubit state for each of 2^n possible settings of the final n-qubit register, with the assistance of $a - n$ additional ancilla qubits, as follows:

$$\begin{aligned} U_L|0\rangle^{\otimes(a-n)}|0\rangle^{\otimes n}|i\rangle &= |0\rangle^{\otimes(a-n)}|\psi_i\rangle \\ U_R|0\rangle^{\otimes(a-n)}|0\rangle^{\otimes n}|j\rangle &= |0\rangle^{\otimes(a-n)}|\phi_j\rangle\,, \end{aligned} \tag{17.3}$$

where the $2n$-qubit states $|\psi_i\rangle$ and $|\phi_j\rangle$ are chosen such that $\langle\psi_i|\phi_j\rangle = A_{ij}/\alpha$, where A_{ij} is the matrix entry of A in row i and column j—the states $|\psi_i\rangle$ and $|\phi_j\rangle$ encode the (normalized) rows of A and norms of those rows, respectively. For this construction, the normalization constant α satisfies

$$\alpha = \|A\|_F\,,$$

where $\|\cdot\|_F$ is the Frobenius norm. Note that for an $N \times N$ matrix the Frobenius norm satisfies $\|A\| \leq \|A\|_F \leq \sqrt{N}\|A\|$—thus, the value of α achieved by this method can be larger than its minimal possible value ($\|A\|$) by a factor as large as $\sqrt{2^n}$.[2]

There are several methods of implementing the (controlled) state preparation unitaries U_L and U_R, offering tradeoffs between various metrics, as discussed in Section 17.2 on state preparation. Of particular relevance is the T-count and T-depth of the circuit when it is decomposed into a Clifford + T gate set, as the T gate is the most difficult to implement in many fault-tolerant schemes. A general strategy for implementing U_L and U_R involves constructing binary trees representing the amplitudes in the states $|\psi_i\rangle$ and $|\phi_j\rangle$ in Eq. (17.3), and building the state preparation unitaries out of controlled Y rotations by angles

[2] See [429, Lemma 50] for a variant of this method yielding normalization factor $\alpha = \sqrt{n_q(A)n_{2-q}(A^\dagger)}$ for $q \in [0, 2]$, where $n_q(A) = \max_i \|A_{i,\cdot}\|_q^q$, with $\|\cdot\|_q$ the vector q-norm and $A_{i,\cdot}$ the i-th row of A.

	Optimized for min depth	Optimized for min count
# Qubits	$4N^2$	$N\log(1/\epsilon)$
T-Depth	$10\log(N) + 24\log(1/\epsilon)$	$8N + 12\log(N)(\log(1/\epsilon))^2$
T-Count	$12N^2\log(1/\epsilon)$	$16N\log(1/\epsilon) + 12\log(N)(\log(1/\epsilon))^2$

Table 17.3 Explicit resource counts for block-encoding circuits of arbitrary matrices of classical data. These expressions omit subleading terms; the full expressions can be found in [296].

defined in those binary trees (the controlled Y rotations are performed *approximately* in a discrete gate set like Clifford + T, leading to a nonexact block-encoding). Tradeoffs involving the T-depth, T-count, and number of ancilla qubits are established based on the manner in which these controlled Y gates are performed. For example, the shallowest implementation [296] requires a large number of ancilla qubits, which are used to perform all the controlled Y rotations in one parallel layer, before "injecting" a subset of these ancillas into the main data qubits using a controlled SWAP network and then uncomputing the ancillas.

Dominant resource cost (gates/qubits)

The shallowest implementations are able to achieve polylog(N) depth for U_L and U_R (and hence for U_A), at the expense of $a = O(N^2)$ ancilla qubits. On the other hand, the fewest number of ancillas needed by this family of methods would be $a = n$, in which case the circuit depth would scale as $O(N^2)$. While the total circuit size must be at least $\Omega(N^2)$, the number of gates in the circuit that are T gates can be as small as $O(N)$ using the techniques in [722].

Detailed resource counts (including the constant prefactors for key metrics) and implementations of block-encodings were studied in [296]. We reproduce their resource counts optimized for minimum T-depth and for T-count in Table 17.3.

Caveats

An important caveat is that the total gate complexity of U_A must be at least $\Omega(N^2)$, reflecting the N^2 degrees of freedom in the arbitrary $N \times N$ matrix A. Thus, while A operates on a quantum system of only $\log_2(N)$ qubits, achieving depth polylog(N) requires parallel-acting gates across at least $\Omega(N^2)$ ancilla qubits. In many algorithms, it is assumed that the cost of implementing U_A is polylog(N), in order to preserve an end-to-end runtime that is polylog(N) and claims of exponential speedup. This is only defensible if the key metric is the circuit depth and if many ancilla qubits are available. Furthermore, the normal-

ization constant α can introduce poly(N) factors into an end-to-end analysis, owing to the fact that $\|A\|_F$ can be larger than $\|A\|$ by a $\sqrt{N}$ factor.

Another caveat to note is that if the matrix being block-encoded is sparse and if the values and locations of its nonzero entries can be computed efficiently, or if the matrix enjoys some structure in the data in addition to sparsity, then more efficient block-encoding methods can be employed—see Section 10.1 on block-encoding for details. In those cases, the results stated here may not be applicable.

Example use cases

In financial portfolio optimization, classical data representing average historical returns and covariance matrices for a universe of assets is needed in a quantum algorithm for optimizing a portfolio. See, for example, [328]. Similarly, in quantum machine learning based on quantum linear algebra, the algorithm often requires fast coherent access to large matrices of classically stored data.

Further reading

- An excellent overview of block-encodings and quantum linear algebra: [431].
- A detailed resource count of block-encoding with explicit circuits: [296].
- Select-SWAP QRAM and a tradeoff between qubit count and T gates: [722].
- For sparse matrices of classical data, or matrices expressed as a linear combination of Pauli matrices, more efficient methods for block-encoding exist. Asymptotic resource expressions for varying number of ancilla qubits are reported in [1084].