4

Combinatorial optimization

Combinatorial optimization problems are tasks where one seeks an optimal solution among a finite set of possible candidates. In industrial settings, combinatorial optimization arises via scheduling, routing, resource allocation, supply chain management, and other logistics problems, where it can be difficult to find optimal solutions that obey various desired constraints. The field of operations research—which came to prominence after its application to logistics problems faced by World War II–era militaries—applies methods of combinatorial optimization (as well as continuous optimization) to these problem areas for improved decision-making and efficiency in real-world problems.

Combinatorial optimization problems are also at the heart of classical theoretical computer science, where they are used to characterize and delineate complexity classes, such as P and NP. Typical combinatorial optimization problems have limited structure to exploit, and therefore quantum computing most often only provides polynomial speedups. In fact, it came as a surprise in the early days of quantum computing research that for a wide variety of such problems, quantum computers do offer up to quadratic speedups via Grover's search algorithm [464]. Subsequently, much effort was devoted to understanding how Grover search and its generalization, amplitude amplification, offer speedups for various combinatorial optimization problems.

In this chapter, we cover several distinct approaches to solving combinatorial optimization problems. First, we look at combinatorial optimization through its relation to search problems, where Grover's algorithm, or its generalizations, can be applied to give a quadratic or subquadratic speedup. Then, we cover several proposals—variational algorithms (viewed as an exact algorithm), the adiabatic algorithm, and the "short-path" algorithm [505, 329]—that have the potential to surpass the quadratic speedup of Grover's algorithm. We discuss the (limited) evidence that these approaches could generate significant advantages, as well as the associated caveats.

We do not specifically cover the large body of work on quantum approaches for *approximate* combinatorial optimization (typically variational quantum algorithms or quantum annealing). These algorithms usually translate the optimization problem to energy minimization of a spin system with a Hamiltonian that encodes the classical objective function. They apply some physically motivated heuristics to efficiently generate solutions that have low energy, and seek a better objective value than could be generated classically in a comparable amount of time. An advantage of these approaches is that they are often more compatible with noisy near-term hardware. While approximate optimization remains an interesting direction, these quantum algorithms are heuristic and there is a general scarcity of concrete evidence that they will deliver practical advantages.

We refer the reader to [6] for a comprehensive survey of quantum methods for combinatorial and continuous optimization.

The authors are grateful to Ashley Montanaro for reviewing this chapter.

4.1 Search algorithms à la Grover

Overview

Grover's search algorithm [464] and its generalizations, such as amplitude amplification, are essential sources of quantum speedups. A straightforward application of Grover search in the spirit of optimization is quantum minimum finding [367, 48], which provides a quadratic speedup for finding the minimizer of a function on a given set of elements.

As search is a generic primitive, Grover's algorithm is widely applicable, and it can speed up many subroutines, especially in algorithms for combinatorial optimization. We list a few representative applications that demonstrate how Grover's algorithm may be applied to speed up combinatorial optimization.

Actual end-to-end problem(s) solved

The goal is to solve a search problem, that is, decide whether there is an element among a set of objects that satisfies some criterion, and if there is such an object, find one. Many combinatorial optimization problems are fundamentally search problems; a notable class of examples are graph problems, such as finding a maximal independent set, a k-coloring, a lowest weight Hamiltonian

cycle[1] (called the traveling salesperson problem), or the shortest path between two vertices.

For conceptual clarity, here, we focus on the prototypical Boolean satisfiability problem, that is, SAT solving: given a Boolean formula in the so-called *conjunctive normal form*, decide whether it has a satisfying Boolean assignment (and if so, find one). A formula in this form consists of some constraints (called *clauses*) each containing the logical AND of some variables or their negation (called *literals*). We denote the number of Boolean variables by n and the total number of literals of the formula by ℓ (typically $\ell \geq n$ since each variable should appear at least once).

Dominant resource cost/complexity

If there are at least m marked elements among N possible ones, then the search problem can be solved with high probability by using $O(\sqrt{N/m})$ Grover iterations. Each Grover iteration requires generating a uniform superposition over the N elements starting from the all $|0\rangle$ state and checking whether an element is marked (in superposition), which can be implemented with gate cost $O(\ell+n)$. If the formula is satisfiable, then there is at least one solution, thus $O(\sqrt{2^n})$ Grover iterations suffice, giving an overall complexity of $O((\ell + n)\sqrt{2^n})$.

In some applications, it is useful to consider a generalization of Grover search, amplitude amplification, which enables working with an arbitrary prior distribution on the elements, unlike Grover's algorithm which effectively uses a uniform prior. The relevance of this extension can be seen through the example of 3-SAT, which is a restricted version of SAT where each clause has at most 3 literals. A clever application of amplitude amplification described by Ambainis [24] for solving 3-SAT more efficiently uses Schöning's algorithm [908] and thus generates a nontrivial prior distribution on the solutions.

The complexity of amplitude amplification is similar to that of Grover search in general. If $|\psi\rangle$ is the quantum state representing the prior distribution, so that measuring the state yields a marked element with probability at least p, then $O(\sqrt{1/p})$ "Grover iterations" suffice to find a marked element with high probability. The algorithm requires preparing the initial state $|\psi\rangle$, and then each iteration consists of a reflection $2|\psi\rangle\langle\psi| - I$ around $|\psi\rangle$ and checking whether an element is marked (in superposition). The former reflection can be implemented with two uses of the circuit that prepares $|\psi\rangle$ from the all $|0\rangle$ state, and a reflection about the all $|0\rangle$ state.

[1] A Hamiltonian cycle in a graph is a cycle that visits each vertex once, not to be confused with a quantum Hamiltonian.

Existing resource estimates

There are several studies on the resource estimation of Grover-type (sub)quadratic speedups. Due to the wide range of these problems, we do not focus on explicit gate counts on any particular problem/implementation variant, but rather list some prominent articles and illustrate their findings at a high level [223, 895, 79, 217, 216, 532]. Unfortunately, these recent studies revealed that quadratic or smaller speedups alone are unlikely to be useful for the foreseeable future, unless the large overheads of current fault-tolerant quantum computing schemes can be greatly reduced. For example, [223] concluded that even if there is some reasonable advantage in quantum gate counts for solving the constraint satisfaction problems that they consider, the classical computation supporting the fault-tolerant quantum computation actually voids the speedup in practice. They state that "Even when considering only problem instances that can be solved within one day, we find that there are potentially large quantum speedups available. ... However, the number of physical qubits used is extremely large, ... In particular, the quantum advantage disappears if one includes the cost of the classical processing power required to perform decoding of the surface code using current techniques." The most recent of the references listed above [532] estimates that achieving a quantum advantage via a quadratic speedup requires at least a month-long computation already if each iteration contains at least one floating-point operation. The situation looks more promising for cubic and quartic speedups, but unfortunately such improvements seem to require techniques beyond Grover search.

Caveats

Grover originally described his result as "A fast quantum mechanical algorithm for database search" [464]. If we work in the circuit model of quantum computation, then strictly speaking Grover search gives a slowdown for database search, as every Grover iteration needs to "touch" every element in the database. If we anyway need to touch all N elements in the database, then the best we can do is to simply go over every element in linear time $O(N)$. Grover's search circuit, on the other hand, would have gate complexity $\widetilde{O}(N^{3/2})$, clearly worse than sequentially going through the entire dataset.

In the database scenario, we can only recover the quadratic speedup if we assume that we can use a quantum random access memory (QRAM), with constant (or logarithmic) cost for each database query. The analogous assumption regarding ordinary RAM is often made in classical computer science, simply because RAM calls are cheap in practice. However, since a RAM call should be able to touch every bit of the database, from a circuit complexity perspective

a RAM call must have gate cost at least N. On the other hand, from a time complexity perspective, one can view a RAM call as a massively parallel piece of computation implementable in a binary tree structure with logarithmic depth.[2] While QRAM can also be implemented with a quantum circuit of $O(\log(N))$ depth, a similar accounting might not be fair in the quantum case, depending on the eventual cost of hardware implementation—especially if error correction of the QRAM is necessary and the entire QRAM circuit is implemented in a fault-tolerant fashion.

Nevertheless, Grover's algorithm can provide a quadratic speedup without extra hardware assumptions when the elements of the list that we search over can be easily generated and checked "on the fly." For example, in the case of SAT, we search over the 2^n possible truth assignments, yet we can easily check whether an individual assignment is satisfactory by simply substituting the assignment into the formula and evaluating the resulting Boolean expression. This is the defining feature of problems in the complexity class NP, whose solutions are efficient to verify.

Comparable classical complexity

For the unstructured search problem, exhaustive search is essentially the best that can be done, with a running time $\sim \ell \cdot 2^n$. Of course, SAT seems to be far from unstructured, but under the Strong Exponential-Time Hypothesis [558, 220] the best classical algorithm for SAT has running time $2^{n-o(n)}$.

A similar argument holds for the generalized problem considered in the setting of amplitude amplification: if we have some prior distribution, we can classically find a marked element by sampling from this distribution roughly $1/p$ times. Since unstructured search is a special case of this problem, we cannot hope for a better classical algorithm in general.

Speedup

The speedup is quadratic in terms of the number of required iterations if we compare to corresponding naive classical algorithms. It can be shown that this speedup is optimal in the black-box query model [122]. Moreover, we do not expect that there would be a bigger than quadratic speedup in gate complexity [210] in the general (non-black-box) case.

[2] Viewing RAM as a low-depth circuit disregards issues regarding signal transmission. Considering that the speed of light is finite and we have only 3 dimensions to fit the memory cells into, a RAM call should asymptotically cost at least $\sqrt[3]{N}$ time. In fact, state-of-the-art clock speeds are already in a regime where the speed of light may be a bottleneck, so we might eventually need to reconsider how the time complexity of RAM is modeled.

Outlook

We have discussed how Grover search provides a quadratic speedup for SAT, and how amplitude amplification yields a quadratic speedup for Schöning's 3-SAT algorithm [908]. Since the best known 3-SAT solvers [526, 497] have complexity $O(1.308^n)$—only slightly better than Schöning's $O(1.334^n)$ complexity—this implies a close-to-quadratic quantum speedup. However, note that this relates to worst-case complexity, and on practical instances, the scaling can be much better.

We now comment on some of the other combinatorial optimization problems where Grover's algorithm can be used as a subroutine. One class of examples is graph-related problems. In the literature, these problems are most often studied in the query model, therefore, here we also only discuss their speedup in terms of query complexity. (Since these are (sub)quadratic speedups, we know that the fault-tolerant resource estimates will be unfavorable anyway, as discussed above.) For instance, the problem of finding the shortest paths from a single source s in graph $G = (V, E)$ to all other vertices $v \in V$ can be solved classically using Dijkstra's algorithm in time $O(|E| + |V| \log |V|)$ if the graph is provided with its adjacency list (and with query complexity $O(|E|)$), whereas the quantum query complexity of this problem is $\widetilde{\Theta}(\sqrt{|V||E|})$ [368]. Reference [368] determines the query complexity of several other graph problems such as deciding graph connectivity and strong connectivity as well as finding the minimum-weight spanning tree. For all of these problems, there is a similar (sub)quadratic quantum speedup.

One graph problem that is often mentioned in connection to quantum computation is the (in)famous traveling salesperson problem. However, for this problem, the best provable speedup is only subquadratic. The naive classical algorithm runs in time $\widetilde{O}(n!)$, and Grover's algorithm offers a quadratic speedup over it. The best classical algorithm uses dynamic programming and runs in time $\widetilde{O}(2^n)$. Ambainis et al. [28] showed how to obtain a speedup over this algorithm by combining classical precalculation with recursive applications of Grover's search resulting in time complexity $\widetilde{O}(1.728^n)$ assuming that QRAM calls have unit costs. Considering the overheads coming from the implementation of QRAM and fault tolerance, the traveling salesperson problem seems to be one of the *least* likely candidates to achieve a practical quantum speedup when the nodes have large degree. For bounded degree graphs there is slightly more hope as quantum algorithms with close-to-quadratic speedups have been devised [789] that do not require QRAM.

Finally, let us mention quantum walk algorithms, which can also be viewed as a generalization of Grover search. However, quantum walks are more dis-

tant relatives of Grover search and can only be applied in more specific settings. They can be used for proving many nontrivial speedups in query complexity, however, the resulting algorithms are often not practical due to high space and/or gate complexity overheads, as is the case for the prototypical element distinctness problem. The query reduction is moderate $N \to N^{2/3}$ in the number of elements N, but the corresponding quantum algorithm [25] unfortunately uses a QRAM consisting of roughly $N^{2/3}$ registers; moreover, the QRAM must be able to store data in superposition.

There are nevertheless more practical quantum walk algorithms applicable, for example, to speed up backtracking algorithms [775, 27, 569, 743], which are among the most successful and widely used classical heuristics for solving SAT instances in practice. The quantum algorithm can achieve an essentially quadratic speedup compared to its classical backtracking variant. This approach is applicable to the traveling salesperson problem in the special case that the graph has degree at most 4 [789]. For resource estimates, see the earlier quoted reference [223]. A further extension of this algorithm is applicable to branch-and-bound algorithms [776, 247], and in some cases yields running times that are substantially better than what we know can be achieved by naively using Grover's algorithm. For example, it can find exact ground states for most instances of the Sherrington–Kirkpatrick model [933] in time $O(2^{0.226n})$ [776], which means about a quadratic speedup compared to classical methods. Branch-and-bound-based speedups can also be applied to solve mixed-integer programs, which include certain formulations of the portfolio optimization problem [247].

There is a plethora of other applications of quantum search speedups, ranging from machine learning [1040] to dynamic programming solutions of other NP-hard problems [28], which we do not discuss here for length constraints and due to discouraging resource estimates for (sub)quadratic quantum speedups.

4.2 Beyond quadratic speedups in exact combinatorial optimization

Overview

The discovery of Grover's algorithm [464] (later generalized to amplitude amplification) has long been the source of enthusiasm that quantum algorithms can be advantageous for combinatorial optimization, as it leads to quadratic asymptotic speedups for many concrete end-to-end search problems in this area. However, resource estimates indicate that early and intermediate-term

fault-tolerant devices will fail to deliver practical advantages when the available speedup is only quadratic, due to intrinsic overheads of quantum computation compared to classical computation (see, e.g., [223, 79]). Thus, identifying whether beyond-quadratic speedups are available is of principal importance for identifying end-to-end practical advantages in combinatorial optimization. Despite the fact that Grover's algorithm is optimal in the black-box (unstructured) setting, superquadratic speedups could be possible when the combinatorial optimization problem has a certain structure that can be better exploited by a quantum algorithm than a classical algorithm.

Unfortunately, many proposals that could conceivably deliver superquadratic speedups lack rigorous theoretical performance guarantees. This includes the quantum adiabatic algorithm and variational quantum algorithms such as the quantum approximate optimization algorithm (QAOA) [384], which is typically formulated to give approximate solutions, but at higher cost could also be used to find exact solutions. Limited analytic and numerical work provides some evidence (e.g., [179, 928]) that QAOA could outperform a vanilla application of Grover's algorithm to the k-SAT problem, but provides no definitive conclusion on the matter. Alternatively, a line of work in [505, 329] studies a different algorithm (related in certain aspects to the quantum adiabatic algorithm) and provides rigorous running time guarantees that *slightly* surpass Grover's algorithm.

However, while these algorithms may have a speedup over Grover's algorithm, this does not entail a superquadratic speedup over the *best* classical algorithm, which can often exploit structure in other ways to do much better than exhaustive search. Overall, it remains an open question whether quantum algorithms can provide superquadratic speedups for useful problems in exact combinatorial optimization.

Actual end-to-end problem(s) solved

Combinatorial optimization problems ask to find which solution is optimal among a finite set of possible candidates. Here, we stick to binary optimization on n bits, where the universe of possible candidates are bit strings $z = (z_1, z_2, \ldots, z_n) \in \{1, -1\}^n$. The input to the problem is a compact description of some cost function $C : \{1, -1\}^n \to \mathbb{R}$, and the desired output is the string z^* for which C is minimized. Let $E^* = C(z^*)$ denote the optimal value of the cost function. For simplicity we assume z^* is unique and E^* is known ahead of time.[3] This setting contrasts with that of *approximate* optimization, where the

[3] This assumption can often be relaxed at the expense of at most poly(n) overhead, for example, by iterating over all possible values E^* might take, which fall within a poly(n)-size range when the cost function consists of only poly(n) constant-size (integer-valued) terms.

acceptable outputs include a much larger set of strings z that are not necessarily optimal solutions, but are still good enough, for example, because they achieve a nontrivial approximation ratio $|C(z)|/|E^*|$ with the optimal cost value. Classical and quantum algorithms for approximate optimization are often heuristic, making it more difficult to systematically study the complexity of the algorithms and the possibility that quantum algorithms may provide a speedup.

Concrete examples can be formed by choosing the function $C(z)$ to be a low-degree polynomial in the bits of z. For example, if C is a degree-2 polynomial in z, this is a quadratic unconstrained binary optimization (QUBO) problem, which is also equivalent to the classical Ising spin model from Eq. (1.3). If, furthermore, every term of C has degree exactly 2 (no degree-1 or constant terms) and every coefficient is either 0 or 1, then the problem is equivalent to a MAX-CUT problem. Finally, if C is a sum of degree-3 terms of the form

$$z_a z_b z_c + z_a z_b + z_a z_c + z_b z_c + z_a + z_b + z_c \,,$$

where

$$z_a, z_b, z_c \in \{z_1, -z_1, z_2, -z_2, \ldots, z_n, -z_n\} \,,$$

then the problem is equivalent to a MAX-3-SAT instance in conjunctive normal form. To see this, note that if $z_a = z_b = z_c = 1$, the term evaluates to 7, and for any other setting, it evaluates to -1. Thus, the solution z^* that optimizes C represents the bit string that minimizes the number of "unsatisfied" clauses for which $z_a = z_b = z_c = 1$. This is easily generalized from MAX-3-SAT to MAX-k-SAT.

For a fixed instance C, the quantum algorithms must find z^* with high probability over measurement outcomes. If it does so for every C chosen from some class of problem, we say it succeeds in the worst case. Alternatively, we can consider ensembles of instances chosen from some class of problem; if for a large fraction of instances from the ensemble, the algorithm finds z^* with high probability, then we say the algorithm succeeds in the average case.[4] A commonly considered average-case ensemble is the Sherrington–Kirkpatrick (SK) model [933], defined as

$$C(z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} J_{ij} z_i z_j \quad \text{where} \quad J_{ij} \sim \mathcal{N}(0,1), \tag{4.1}$$

[4] A more typical definition of the average-case complexity of an algorithm is the expected runtime required for it to find the solution z^*, averaged over both choice of instance and internal algorithmic randomness (i.e., classical coin flips or quantum measurement outcomes). This definition is related to the convention we follow, but it is more coarse grained as it does not distinguish between the two types of randomness, the latter of which can be boosted by repetition.

where the coefficients J_{ij} are drawn randomly from a standard Gaussian distribution $\mathcal{N}(0,1)$. The SK model is relevant in spin glass theory, and can be generalized to higher-degree interactions, where it is referred to as the p-spin model [345]. Another ensemble is the random MAX-k-SAT ensemble, where MAX-k-SAT instances are generated by choosing each clause uniformly at random with some fixed clause-to-variable ratio (see, e.g., [307]).

Dominant resource cost/complexity

A vanilla application of Grover's algorithm to binary optimization problems achieves $O^*(2^{0.5n})$ running time, where notation $O^*(2^{an})$ is shorthand for $\text{poly}(n)2^{an}$. We cover three approaches to solving binary optimization problems on a quantum computer that have some potential to improve upon this running time. Note that all of these algorithms require polynomial (in fact, linear $O(n)$) space. However, their running time is expected to scale exponentially in n.

- First, we consider variational quantum algorithms, using the QAOA [384] as a representative. These algorithms are typically studied as efficient (polynomial-time) quantum algorithms that produce approximate solutions, that is, strings $z \neq z^*$ for which $C(z)$ is small, but not optimal. However, they may also be viewed as exact algorithms, since, if repeated a sufficient number of times, they eventually produce the exactly optimal z^*. The QAOA fixes a depth parameter p and variational parameters $\gamma = (\gamma_1, \ldots, \gamma_p)$ and $\beta = (\beta_1, \ldots, \beta_p)$ (sometimes these are set to some fixed instance-independent value, and sometimes they are variationally updated on subsequent repetitions of the algorithm). The QAOA starts in the n-qubit equal superposition state $|+\rangle^{\otimes n}$ and implements alternating rounds of rotations about the diagonal cost function C and a "mixing" operator $X = \sum_i X_i$, where X_i denotes the Pauli-X gate about qubit i. The state produced by QAOA is thus given by

$$|\psi_{\gamma,\beta}\rangle = \mathrm{e}^{-\mathrm{i}\beta_p X}\mathrm{e}^{-\mathrm{i}\gamma_p C}\cdots\mathrm{e}^{-\mathrm{i}\beta_2 X}\mathrm{e}^{-\mathrm{i}\gamma_2 C}\mathrm{e}^{-\mathrm{i}\beta_1 X}\mathrm{e}^{-\mathrm{i}\gamma_1 C}|+\rangle^{\otimes n}.$$

  If one makes a computational basis measurement of $|\psi_{\gamma,\beta}\rangle$, one obtains z^* with probability $|\langle z^*|\psi_{\gamma,\beta}\rangle|^2$. The expected number of repetitions required to obtain z^* is the inverse of this probability, and this running time can be quadratically sped up by performing amplitude amplification on top of the QAOA protocol; thus, the QAOA unitary is applied $O(|\langle z^*|\psi_{\gamma,\beta}\rangle|^{-1})$ times. Implementing the QAOA unitary typically requires only $p \cdot \text{poly}(n)$ gates, as each of the rotations about X and C are efficient to implement. For hard combinatorial optimization problems such as typical MAX-k-SAT instances, the

expectation is that the total running time required will be exponential. If the depth p is chosen to be constant or even poly(n), the dominant cost will come from the $O(|\langle z^*|\psi_{\gamma,\beta}\rangle|^{-1})$ repetitions required to amplify the $|z^*\rangle$ state. Alternatively, one can reduce the number of repetitions needed to $O(1)$ at the expense of taking p to be very large (at least exponentially large in n); indeed, for sufficiently large p, the QAOA can be viewed as a Trotterized simulation of the adiabatic algorithm [384].

There is some analytic evidence that the QAOA may outperform Grover's algorithm at finding the exact solution for constant p in certain cases. Reference [179] studied the QAOA applied to hard (i.e., near the satisfiability threshold) k-SAT instances with instance-independent choice of γ, β for constant p, and developed an analytic formula for the expected success probability $|\langle z^*|\psi_{\gamma,\beta}\rangle|^2$ averaged over random instance in the limit $n \to \infty$. This formula was evaluated numerically and suggested, for example, that the average success probability behaves as $2^{-0.33n}$ for $p = 10$ on 8-SAT. One might be tempted to declare that this implies an overall average running time of $O^*(2^{0.33n/2})$, substantially better than Grover, but such a conclusion is not analytically supported as the average of the inverse probability can be much larger than the inverse of the average probability. Nevertheless, it provides intriguing evidence in favor of such a conclusion. Further numerical evidence that QAOA may be effective as an exact algorithm was provided in [928], which numerically assessed the performance of QAOA on instances of the low autocorrelation binary sequences (LABS) problem up to $n = 40$, although compared to the best classical heuristic solver, the advantage appeared to be subquadratic.

- Second, we consider the quantum adiabatic algorithm [382, 16]. The standard approach, as applied to binary optimization problems, is to start in the state $|+\rangle^{\otimes n}$ and evolve by a Hamiltonian that interpolates along a path $H(s)$ parameterized by $s \in [0, 1]$, given by

$$H(s) = (1 - s)(-X) + sC\,. \tag{4.2}$$

It is important to note that the ground state of $H(0)$ is $|+\rangle^{\otimes n}$ and the ground state of $H(1)$ is $|z^*\rangle$. This evolution can be simulated on a fault-tolerant gate-based quantum computer using Hamiltonian simulation, and its running time is dominated by the inverse of the minimum spectral gap $\Delta_{\min}$ of $H(s)$. That is, the gate complexity to run the algorithm and produce $|z^*\rangle$ scales as at least $\Delta_{\min}^{-1}$ and possibly a larger power of $\Delta_{\min}^{-1}$. Much numerical work has been done on the performance of the adiabatic algorithm on small instances of combinatorial optimization problems, but it generally lacks analytical guarantees. The expectation is that $\Delta_{\min}$ will be exponentially small

[638, 1071, 517] in n (or worse, see, e.g., [23, 1032]), meaning the running time of the algorithm is exponentially large, but it remains possible that it surpasses the $O^*(2^{0.5n})$ running time of Grover's algorithm in some cases, and could in principle deliver a superquadratic speedup.

- Third, we consider the *short-path* algorithm studied in [505, 506, 508] and a dual version of the algorithm studied in [329]. The goal of these algorithms was to be able to provide a rigorous guarantee that the algorithm can find z^* in time $2^{(0.5-c)n}$ for some value of $c > 0$. Similar to the adiabatic algorithm, the short-path algorithm also considers a single-parameter family of Hamiltonians

$$H(s) = (1-s)f_X\left(-\frac{X}{n}\right) + sf_Z\left(\frac{C}{|E^*|}\right), \tag{4.3}$$

where $f_X, f_Z : \mathbb{R} \to \mathbb{R}$ are monotonic filter functions, and each term X/n and $C/|E^*|$ are normalized to have minimum value -1. The idea of the short-path algorithm is to, rather than evolve smoothly from $s = 0$ to $s = 1$, perform a pair of discrete "jumps." The first jump goes from the ground state $|+\rangle^{\otimes n}$ at $s = 0$ to the ground state $|\psi_b\rangle$ of an intermediate point with $s = b$. The second jump goes from $|\psi_b\rangle$ to the ground state $|z^*\rangle$ at $s = 1$. The jumps are accomplished with quantum phase estimation (or more advanced versions utilizing the quantum singular value transformation) of the Hamiltonian H_b combined with amplitude amplification. The running time of the algorithm is [329, Theorem 1]

$$\text{poly}(n) \cdot \frac{1}{\Delta} \cdot \left(\frac{1}{|\langle +|\psi_b\rangle|} + \frac{1}{|\langle \psi_b|z^*\rangle|}\right), \tag{4.4}$$

where Δ is the spectral gap of the Hamiltonian $H(b)$. The Δ^{-1} factor comes from the need to perform phase estimation at $O(\Delta)$ resolution to successfully prepare $|\psi_b\rangle$, and the two additive inverse overlap terms represent the number of rounds of amplitude amplification for the first and second jumps, respectively. In [505], filter functions $f_X(x) = x^K$ for odd integers K (e.g., $K = 3$) and $f_Z(x) = x$ were chosen, and b was chosen close to 1, such that the first term of Eq. (4.3) could be viewed as a small perturbation of the second term. If C is an instance of MAX-Ek-LIN2, that is, if it is a polynomial for which all monomials are degree exactly k, then it was shown that certain conditions on the spectral density of C near the optimal cost value imply sufficient analytic control of Δ and the other parameters in Eq. (4.4) such that the algorithm runs in time $O^*(2^{(0.5-c)n})$ for $c > 0$. However, it remained unclear when these conditions were met. Inspired by [505], [329] proposed using the filter functions $f_X(x) = x$ and $f_Z(x) = \min(0, (x + 1 - \eta)/\eta)$ for a fixed choice of $\eta \in [0, 1]$, and chose a value of s close to 0 (rather than

close to 1). In this sense, the algorithm in [329] is dual to that of [505]. These modifications allowed additional statements to be proved. For example, it was unconditionally shown that the algorithm solves k-SAT (whether or not a formula has a fully satisfiable solution) in time upper bounded by $O^*(2^{(0.5-c)n})$ for a (extremely small) constant $c > 0$, and that the same is true for typical instances of the SK model and its higher-body generalization (p-spin model), a polynomial speedup over Grover's algorithm and superquadratic advantage over classical exhaustive search.

Existing resource estimates

Reference [895] compiled resource estimates for various primitive tasks related to combinatorial optimization. For example, it estimated that for an $n = 512$ instance of the SK model, implementing a single QAOA step $e^{-i\beta_j X} e^{-i\gamma_j C}$ would require 577 logical qubits and 5.0×10^5 Toffoli gates. A similar estimate would hold for performing a single step of adiabatic evolution with a first-order product formula. The total logical estimate for finding z^* would be the product of the depth of the circuit and any number of repetitions or rounds of amplitude amplification. An estimate of the physical resource cost could then be computed for a specific fault-tolerant architecture. Without knowing the number of repetitions, it is hard to give precise estimates, but a rough attempt was made in [79] for different speedup factors. There, under different possible assumptions on the amount of classical parallelism available, a breakeven point was estimated for different possible polynomial speedups (quadratic, cubic, and quartic). It was found that with a quartic speedup, the breakeven point could be reasonable (on the order of seconds to hours) even assuming the availability of classical parallelism.

Caveats

There are several caveats. The most salient one is that for most of the algorithms above, there is no provable beyond-Grover advantage. Meanwhile, in the case of [329], the size of the provable beyond-Grover advantage is miniscule. The prospect of these algorithms is thus left to extrapolations from numerical simulations carried out at very small instance sizes and speculation based on physical principles.

A second important caveat is that to deliver practical superquadratic speedups, the performance of the quantum algorithm needs to be compared to the best classical algorithm, which is often substantially better than the $O^*(2^n)$ running time of exhaustive enumeration. For example, 3-SAT problems are classically solvable in $O^*(2^{0.39n})$ time [497].

Along these lines, a third caveat is the existence of classical "quantum Monte Carlo" algorithms (see, e.g., [383, 188, 570, 322, 324]), which can, under certain conditions, classically simulate the quantum algorithms described above. This is because the Hamiltonians in Eqs. (4.2) and (4.3) are *stoquastic* Hamiltonians, defined by the property that their off-diagonal matrix elements are non-positive (when written in the computational basis). Stoquasticity implies that the ground state of the Hamiltonian can be written such that all amplitudes are non-negative real numbers [191], meaning that these Hamiltonians avoid the so-called "sign problem" enabling the potential application of quantum Monte Carlo techniques. To be clear, it remains possible that quantum algorithms for these combinatorial optimization problems involving stoquastic Hamiltonians can evade classical simulation—indeed, superpolynomial oracle separations have been shown between classical computation and adiabatic quantum computation restricted to stoquastic paths [510, 432]—but it is something to keep in mind when designing algorithms based on stoquastic Hamiltonians.

A final caveat is that the quantum algorithms described here are typically not amenable to parallelization, although in principle QAOA could be parallelized if one opts not to use amplitude amplification (resulting in worse asymptotic complexity). This lies in stark contrast to many classical optimization algorithms for exact combinatorial optimization which are highly parallelizable, a feature that can be exploited to significantly reduce the running time of these classical algorithms on high-performance computers, making achieving practical quantum advantage more difficult [79].

Comparable classical complexity and challenging instance sizes

For many binary optimization problems, there exist classical algorithms that exploit the structure of the problem to perform significantly better than exhaustive search. For example, the best 3-SAT algorithm runs in time $O^*(2^{0.39n})$ and in general k-SAT can be solved in time $2^{(1-\Omega(1/k))n}$ [497]. This running time suggests the solution will be impractical once n is on the order of 100. The algorithm analyzed in [497] is designed for the worst case, and it is likely not the best practical algorithm for typical instances. For random instances, the hardness of k-SAT depends sensitively on the clause-to-variable ratio α. Remarkably, heuristic algorithms can succeed at finding a satisfiable solution for typical instances with thousands or even tens of thousands of variables even very close to the satisfiability threshold α_c where most instances become unsatisfiable (e.g., [739]). However, these algorithms are expected to fail sufficiently close to the satisfiability threshold and in the worst case.

Similarly, the SK model admits a classical branch-and-bound algorithm guaranteed to run in time $2^{0.45n}$ (for a large fraction of instances) and likely

better than that in practice [776]. However, once the interaction degree becomes larger than 2, the problem becomes significantly harder. The branch-and-bound algorithm is not known to generalize to the p-spin model, and for $p \geq 3$ there is no known classical algorithm that provably achieves $2^{(1-c)n}$ for any constant c (although it has not garnered much attention, see [329]). Similarly, in contrast to k-SAT, the MAX-k-SAT problem (i.e., the version of the problem that asks for the optimal assignment even if it does not satisfy all the clauses) only has a $O^*(2^{(1-c)n})$ time algorithm for $k = 2$, and, notably, this algorithm requires exponential space [1047].

Speedup

As there are generally no rigorous running time guarantees for the quantum algorithms, the speedup cannot be estimated. However, it is worth emphasizing that for hard combinatorial optimization problems, the speedup could be superquadratic, but it is not expected to be superpolynomial.

The rigorous results of [329] establish a beyond-Grover running time, but the only case in which the speedup is beyond quadratic when compared with the best known classical algorithm is the p-spin model with $p \geq 3$ (here, the comparison benefits from little work on classical algorithms for the problem).

We also mention the result of [904], which studies a quantum algorithm for random instances of a QUBO-like combinatorial optimization problem with a "planted" optimal solution—the goal is to exactly or approximately find the planted solution, or alternatively to simply distinguish instances drawn from the ensemble with planted solutions from instances drawn from the ensemble without a planted solution. The algorithm generalizes the tensor PCA algorithm of [511] and gives a *quartic* speedup over its closest classical counterpart, although it is unclear if this speedup can extend to non-planted scenarios as well.

NISQ implementation

The QAOA approach is amenable to NISQ implementation (assuming one opts not to apply amplitude amplification on top of it), since the quantum circuit one needs to implement is fairly shallow depth. In this case, the effect of uncorrected errors in the NISQ device may degrade the performance (and require more repetitions to extract the optimal bit string z^*). Similarly, on a NISQ quantum annealer [591, 16], one could run a noisy version of the quantum adiabatic algorithm and repeat until finding the optimal bit string z^*.

Outlook

In contrast to algorithms for approximate optimization, which are often heuristic but run in polynomial time, algorithms for exact optimization are often more rigorous but run in exponential time. For quantum computers to be impactful for exact combinatorial optimization, we require great advancements in the estimated clock speeds of quantum hardware and the overheads of fault-tolerant quantum computing, or else the development of quantum algorithms that significantly improve upon existing (sub)quadratic Grover-type speedups—either quantitatively (bigger speedups) or qualitatively (e.g., requiring only shallow circuits). Although ideas have been proposed that could potentially deliver such improvements, they either come without provable guarantees, provide only minor superquadratic improvement, or only apply to artificial problems. Much more attention shall be devoted to studying these quantum algorithms and developing new ones if we are to leverage them into actual practical advantages, especially considering the extensive amount of work devoted to developing sophisticated classical algorithms for these problems.