A super-polynomial quantum advantage for combinatorial optimization problems

Niklas Pirnay[1], Vincent Ulitzsch[1], Frederik Wilde[2], Jens Eisert[2,3], and Jean-Pierre Seifert[1,4]

[1]Electrical Engineering and Computer Science, Technische Universität Berlin, 10587 Berlin, Germany

[2]Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany

[3]Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany

[4]Fraunhofer SIT, Rheinstraße 75, 64295 Darmstadt, Germany

Combinatorial optimization—a field of research addressing problems that feature strongly in a wealth of practical and industrial contexts—has been identified as one of the core potential fields of applicability of near-term quantum computers. It is still unclear, however, to what extent variational quantum algorithms can actually outperform classical algorithms for this type of problems. In this work, by resorting to computational learning theory and cryptographic notions, we prove that fault-tolerant quantum computers feature a super-polynomial advantage over classical computers in approximating solutions to combinatorial optimization problems. Specifically, building on seminal work of Kearns and Valiant, we construct special instances of the integer programming problem (which in its most general form is NP-complete) that we prove to be hard-to-approximate classically but give an efficient quantum algorithm to approximate the optimal solution of those instances, hence showing a super-polynomial quantum advantage. This result shows that quantum devices have the power to approximate combinatorial optimization solutions beyond the reach of classical efficient algorithms.

1 Introduction

Combinatorial optimization problems arise in a wealth of industrial contexts [7]: Any problem of scheduling, routing or job allocation can basically be seen as a problem of this type. Given the significance of such problems and motivated by the insight that *quantum computers* may offer substantial computational speedups over classical computers [20, 25], it has long been suggested that quantum computers may actually assist in solving such problems. However, many problems of this kind, are known to be NP-complete. This does mean that it is unlikely that either classical or quantum computers can efficiently solve generic combinatorial optimization problems exactly. In other words, it is not reasonable to expect that such problems can be solved in polynomial time using either type of computing paradigm.

This does not mean that one cannot solve practically relevant instances of those problems up to reasonable system sizes or find good approximations. There is indeed a rich body of literature on both heuristic approaches that work well in practice [14] as well as on a rigorous theory of approximating solutions [31]. For example, enormous instances of the *traveling salesperson problem* have been solved optimally for up to 85.900 vertices [2]. But, there is no hope for an efficient either quantum or classical algorithm that is *guaranteed to find the optimal solution.* This begs the two questions whether quantum computers offer an advantage for combinatorial optimization problems and specifically for *approximating* the solution of combinatorial optimization problems.

This topic is particularly prominently discussed in the realm of *near-term quantum computers* [22], for which full quantum error correction and fault tolerance seem out of scope, but which may well offer computational advantages over classical computers [3, 12]. Indeed, for such devices, algorithms such as the *quantum approximate optimization algorithm* [9] have been designed precisely to solve combinatorial optimization problems of the above mentioned kind. Surely these instances of variational algorithms [6, 19] will not always be able to solve such problems: At best, these algorithms may be able to produce approximate solutions that are better than those found by classical computers. They may also be able to efficiently find good approximations for more instances than classical computers when they are used perfectly. When actually operated in realistic, noisy environments, the performance of quantum devices is further reduced. Indeed, for variational algorithms run on noisy devices, some obstacles have been identified for quantum computers that involve circuits that are deeper than logarithmic [23, 26, 29], obstacles that may well be read as indications that it will be challenging to achieve quantum advantages in the presence of realistic noise levels. Specifically for variational algorithms aimed at tackling classical combinatorial optimization problems that are being cast in the form of minimizing the energy of commuting Hamiltonian terms, further obstructions have been identified. Already at logarithmic depth, there is solid evidence that noise spreads over the entire system, challenging a possible quantum advantage in combinatorial optimization [11]. Some small instances of the problem can even be classically efficiently simulated heuristic performance of quantum approximate optimization prmoses can be encouraging [33]. Then, small noise levels are known to assist in the optimization problem accompanying the variational algorithm [17].

The make-or-break question, therefore, is: What is, after all, the potential of quantum computers for solving combinatorial optimization problems? A simple quantum advantage for exactly solving combinatorial optimization problems may be obtained by reducing the integer factoring problem to 3-SAT and leveraging the advantage of Shor's algorithm [25]. A quantum advantage for *approximating* the solution of combinatorial optimization problems seems much more difficult to obtain. Given the practical importance of such a task and its wide applicability, this poses a highly relevant questions.

In this work, we provide a comprehensive answer to the affirmative: A fault tolerant quantum computer can approximate certain combinatorial optimization problems super-polynomially more efficiently than a classical computer. We build on the work of Kearns and Valiant [15], who have shown the classical hardness of approximating the solution of the so-called *formula coloring problem*, a combinatorial optimization problem which generalizes the *graph coloring problem*. We continue to draw inspiration from Ref. [15] when showing an approximation hardness preserving reduction from the formula coloring problem to *integer programming* (a family of combinatorial optimization problems on which variants of quantum approximation have already been applied to [8]). To prove the super-polynomial quantum advantage, we show the classical approximation hardness for certain formula coloring and integer programming instances that are constructed from the *RSA encryption function* and provide an efficient quantum algorithm for approximating the solutions of those instances up to a polynomial factor. We also formulate the hard-to-approximate formula coloring instances in the optimization problem of minimizing the energy of commuting Hamiltonian terms, connecting our findings to the widely studied field of variational quantum optimization. Since the classical approximation hardness stems from the hardness of inverting the *RSA encryption function* [15], the core of the quantum advantage discovered in this work is ultimately essentially borrowed from, once again, Shor's quantum algorithm [25] for factoring.

The kind of reasoning developed here resembles that of Refs. [18, 21, 27], but in contrast to the problem of PAC distribution learning (or the solution of a classification problem) considered in that work, we here apply the mindset followed there to the problem of approximating solutions to problems in combinatorial optimization. The argument we have put forth compellingly shows that full-scale, fault tolerant quantum computers can indeed perform provably substantially better than classical computers on instances of approximating combinatorial optimization problems, in fact, featuring a super-polynomial speedup. On a higher level, this work displays, once more, how helpful *cryptographic methods* can be in the design of quantum algorithms. To make contact with quantum approximate optimization, we also spell out how the graph coloring problem can be written as a Hamiltonian optimization problem. While the results found here are highly motivating and do show the potential of quantum devices to tackle such practically relevant problems, it remains open to which extent this potential can be unlocked for short variational quantum circuits as they are accessible in near-term quantum computers.

In addition to being a technical result presumably interesting in its own right—showcasing the potential of quantum computers to offer speedups when solving classical combinatorial optimization problems—it also conceptually provides guidance on the question what type of speedups one can

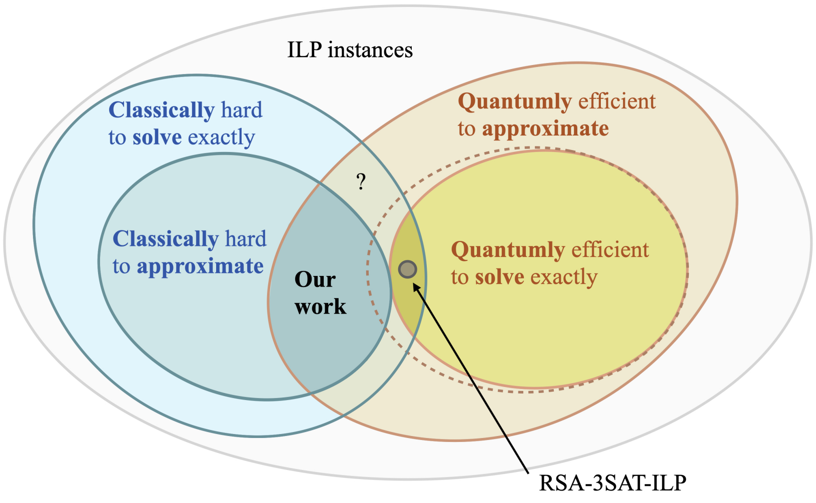

ILP instances
Classically hard to solve exactly
Quantumly efficient to approximate
?
Classically hard to approximate
Our work
Quantumly efficient to solve exactly
RSA-3SAT-ILP

Figure 1: Venn diagram that depicts the sense in which a quantum advantage is proven for the formula coloring and the integer programming problems. The gray set contains all instances of integer linear programming, and the subsets contain the hard or respectively easy to solve instances. By hard to approximate we mean that there is no polynomial time algorithm that approximates the size of the optimal solution up to a factor of $opt^{\alpha} \cdot |I|^{\beta}$, where $|I|$ is the instance size, α, β are constants such that $\alpha \geq 0$, $0 \leq \beta < 1$ and opt denotes the size of the optimal solution. Whether the dotted line holds true, i.e., whether there exists a problem that can be solved *exactly* by a polynomial-time quantum algorithm, but are hard to approximate classically, is left for further research.

expect in any approach. The present work does *not* suggest to solve NP-hard problems exactly on a quantum computer in polynomial time. Instead, we show that a quantum computer can approximate hard instances the hardness of which stems from the hardness of inverting the RSA encryption function in polynomial time, see Fig. 1. This can be seen as a positive result on the potential use of quantum computers to address combinatorial optimization problems. Colloquially speaking, our work provides guidance of the type of speedups one can hope to achieve for this class of problems, on fault tolerant quantum computers and, possibly, also by resorting variational quantum algorithms.

This work is structured in three constituents. First, we introduce preliminary notation and establish a basis for our findings by introducing representation classes, reductions among representations and a framework for representation learning. We here also introduce the formula coloring problem and some basic cryptographic primitives that are key for the understanding of the arguments that follow. The second component of this work focuses on establishing approximation tasks that are provably hard for classical computers. Motivated by this reasoning, we explain the approximation hardness results of Ref. [15] and extend their work by proving an approximation hardness preserving reduction of formula coloring to integer programming. In the third section, we complete the quantum-classical advantage by providing efficient quantum algorithms that solve the classically hard to approximate problems. We finally make a connection to notions of Hamiltonian optimization before concluding this work.

2 Preliminaries

2.1 Notation

For what follows, some notation will be required. We will heavily build on literature from the cryptographic context, and hence make use of substantial notation that is common in this context. By $\{0,1\}^n$ we will denote the set of n-bit strings, whereas $\{0,1\}^*$ are arbitrary finite length bit strings. 2^X is the power set of X, for X being a set. $\mathbb{1}(a)$ is the *indicator function* which equates to 1 if a is true and 0 otherwise. $LSB(x)$ is the least significant bit of x. $\mathbb{Z}_N$ is the residue class ring $\mathbb{Z}/N\mathbb{Z}$. The application of the function $\text{binary}(x_1, \ldots, x_k)$ explicitly converts its inputs $x_1, \ldots, x_k$

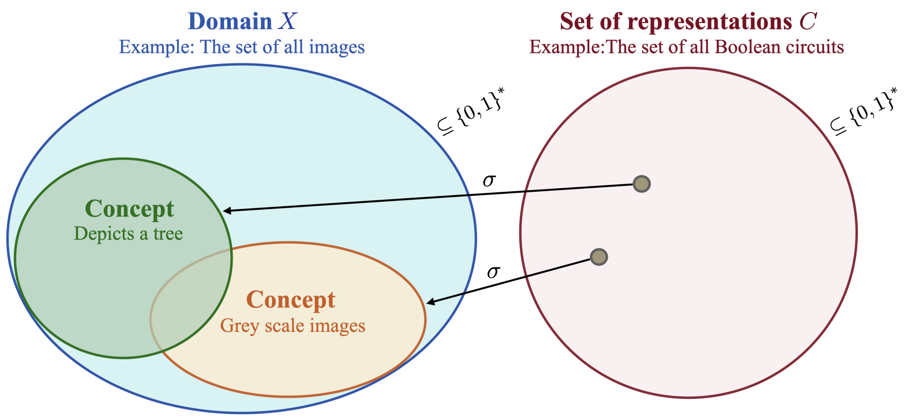


Figure 2: The domain X can formally be seen as a set of finite bit strings. Concepts are subsets of the domain, which can be described by representations $c \in C$. Together with the map σ, mapping representations to concepts, the tuple (σ, C) is called a *representation class*.

to a single coherent bit string using some fixed binary encoding.

2.2 Representation classes

To show a quantum-classical separation for a computational task, one needs a classically hard problem. Many computational tasks that are hard for classical computers may be derived from cryptography, where it can be shown that under *cryptographic assumptions* (such as "factoring is hard") learning certain concepts or properties about a specific cryptographic function is hard. In particular, in this work we are concerned with how these concepts are represented and how large these representation are. To do this, let us introduce the notion of representation classes that capture the model of concepts in a precise manner. Let $X \subseteq \{0,1\}^*$ be a set of binary strings with finite length, called the *domain* which encodes all objects of interest to us. For example, X may be the set of all images or X may be the set of all music songs. A *concept* over X is described by a subset of X which is defined via $\{x \in X \mid \text{concept is true for } x\}$. A concept may be for example "depicts a tree" or "is a happy song".

We are in particular interested in how a concept is represented. Different representations for a concept can, for example, be Figure *Boolean circuits*, *Boolean formulae*, *Turing machines* or *deterministic finite automata* (DFA) [13]. We, therefore, define a *representation class over X* to be the pair (σ, C), where $C \subseteq \{0,1\}^*$ is the set of representation descriptions, for example the set of descriptions for Boolean circuits or finite automata. The function $\sigma : C \to 2^X$ maps a representation description to a concept. For example, σ maps a DFA to the set of bit strings that it accepts or a Boolean formula to its satisfying assignments. We will sometimes denote (σ, C) simply by C if σ is clear from the context. Figure 2 visualizes the relationship between representations and concepts.

Observe that for all $c \in C$, $\sigma(c)$ is a concept over X and the entire image space $\sigma(C)$ is called the *concept class* represented by the representation class (σ, C). We denote by $|c|$ the length of the representation description using some standard encoding. Additionally, for a representation $c \in C$, we denote by $c(x) = \mathbb{1}(\text{if } x \in \sigma(c))$ the *label* of x under the concept $\sigma(c)$, with the index function $\mathbb{1}$. Furthermore, a *labeled sample*

$$S = \{(x, c(x)) \mid x \in \tilde{X} \subseteq X\} \tag{1}$$

of a concept $\sigma(c)$ is a set of *labeled examples* from a subset $\tilde{X}$ of the domain X. Note that a sample consists of multiple examples. Finally, let (ϕ, H) be another representation class over X and let D be a probability distribution over X. For any $h \in H$, we define the error of h under D with respect to a target representation c as

$$\text{error}_{c,D}(h) = \Pr_{x\sim D}[c(x) \neq h(x)]. \tag{2}$$

2.3 Polynomial-time reductions

Polynomial-time reductions are an important building block of this work, as they will be an integral part of our proof of a quantum-classical computational separation for combinatorial optimization problems. Building on the work of Ref. [16], reductions are required to "carry over" classical hardness results of representation learning to combinatorial optimization problems. At the same time, we find that quantum computers can break the construction and lead to a quantum advantage for combinatorial optimization. Let us now introduce the notion of general polynomial-time reductions among computational problems.

Let A, B be two computational problems. Consider the function τ to map an instance $\mathcal{A}$ of A to an instance $\tau(\mathcal{A})$ of B. Furthermore, let g be a function that maps from the solution space of B to the solution space of A. The pair of functions (τ, g) is a *polynomial-time reduction* from A to B, if τ, g are computable in polynomial time and if $y_{\mathcal{B}}$ is a solution of $\tau(\mathcal{A})$ if and only if $g(y_{\mathcal{B}})$ is a solution of A.

Note that while τ maps instances from A to B, g works in the backwards direction, mapping solutions of B to A. This will be important for reductions between combinatorial optimization problems. In some cases, where g is the identity, we call the reduction simply by the instance transformation τ. Further we denote by $A \leq_p B$ ("A polynomial-time reduces to B"), if there exists a polynomial-time reduction from A to B. It is important to note that since the run time of τ and g are at most polynomial in their inputs, the outputs can be larger than the inputs at most by a factor of $\mathrm{poly}(|\mathcal{A}|)$, $\mathrm{poly}(|y_{\mathcal{B}}|)$, respectively.

2.4 Reductions among representations

To understand our proof of the quantum advantage in combinatorial optimization, we require polynomial-time reductions among the evaluation problem of representation classes. Intuitively, these reductions show that one representation class is at least as powerful as another and that they can be transformed into each other. In Ref. [15], these reductions have been used to derive (classical) computationally hard problems for different representations. First we define a technical construction, the *evaluation problem.*

Definition 2.1 (Evaluation problem $\mathit{Eval}(C)$)**.**
Instance *The pair (c, x), where C is a representation class over the domain X, $c \in C$ is a representation description and $x \in X$.*
Solution *The result $c(x)$ of c on x.*

Let $n \in \mathbb{N}$ and define BC_n to be the representation class of *polynomially evaluatable Boolean circuits* with domain $X = \{0,1\}^n$ and with depth $O(\log(n))$ and size $O(\mathrm{poly}(n))$, and let $\mathrm{BC} = \cup_{n\geq 1} \mathrm{BC}_n$. In a similar manner, define BF to be the representation class of *Boolean formulae of poly-size*, define LSTM to be the representation class of *log-space Turing machines* and finally, define DFA to be the representation class of *deterministic finite automata* [13] of poly-size. It holds that

$$\mathit{Eval}(\mathrm{BC}) \leq_p \mathit{Eval}(\mathrm{BF}), \tag{3}$$

$$\mathit{Eval}(\mathrm{BF}) \leq_p \mathit{Eval}(\mathrm{LSTM}), \tag{4}$$

$$\mathit{Eval}(\mathrm{LSTM}) \leq_p \mathit{Eval}(\mathrm{DFA}). \tag{5}$$

Subsequently, we sketch the proof ideas for the three reductions above. The full proofs can be found in Refs. [16] and [15]. From here on after, we consider n to be the size of the input to a Boolean circuit in BC.

(3) We denote this polynomial-time reduction by τ_1. Recall that τ_1 is the instance transformation algorithm and the solution transformation is the identity. Let c be a Boolean circuit in BC with depth $d = O(\log(n))$ and size $s = O(\mathrm{poly}(n))$. Every Boolean circuit can be identified with a directed acyclic graph where each vertex has fan-in at most 2. The instance transformation in the reduction goes by starting at the output vertex of c and recursively building the Boolean formula f by walking back through c and substituting clauses in f. f will then consist of at most 2^d clauses over n variables, which is size $O(\mathrm{poly}(n))$. Clearly, c

and f compute the same function, the reduction (3) holds, since the transformation can be performed by an $O(\text{poly}(n))$-time algorithm. We have $\tau_1((c,x)) = (f,x)$.

(4) We denote this polynomial-time reduction by τ_2. This reduction uses the fact that we can transform any Boolean formula f to a log-space Turing machine m that, on input x computes $m(x) = f(x)$, in time $O(\text{poly}(n))$. The details for this transformation can be found in Ref. [16]. Again, we denote the operation of this instance transformation algorithm as $\tau_2((f,x)) = (m,x)$.

(5) We denote this polynomial-time reduction by τ_3. The reduction uses a transformation of log-space Turing machines to *deterministic finite automata* [13]. In particular, for each log-space Turing machine m, one can construct a DFA t that on input of polynomially many copies of the original input x simulates m [16]. Note that in the reduction here, the input is transformed such that x is repeated $p(n)$ many times and then taken as the input to t, where p is a polynomial in n. We thus have $\tau_3((m,x)) = (t, \underbrace{x,\ldots,x}_{p(n)\text{ times}})$, such that

$$m(x) = t(x,\ldots,x). \tag{6}$$

Sometimes we are only interested in the transformation of the representation description and not in the input x. If we say that we transform a representation description c using $\tau_{1,2,3}$, we omit the second input x and simply write $\tau_{1,2,3}(c)$. Recall that in the reductions above, since the instances are transformed by polynomial-time algorithms, the output instances can be larger than the input at most by a polynomial factor.

2.5 Learning of representations

To obtain a classical hardness result for approximation tasks, the work of Ref. [15] use the so-called *Occam learning framework* [5]. Generally speaking, the Occam learning framework makes a connection between nearly minimal hypotheses which are consistent with observations and the ability to generalize from the observed data in the sense of PAC learning. To introduce this formalism, let $(\sigma, C), (\phi, H)$ be two representation classes over the domain $X \subseteq \{0,1\}^n$. In the following we write C for (σ, C) and H for (ϕ, H) and denote the two representation descriptions $c \in C$ and $h \in H$ as elements of the set of representation descriptions of (σ, C) and (ϕ, H). Given a labeled sample

$$S = \{(x_1, c(x_1)), \ldots, (x_m, c(x_m))\} \tag{7}$$

of m *examples*, we say that $h \in H$ is *consistent* with S, if and only if $c(x_i) = h(x_i)$ for all $i = 1, \ldots, m$. The $x_1, \ldots, x_m \in X$ might be drawn at random according to a distribution D over X. Importantly, we denote by $opt_{Con}(S)$ the size of the smallest $h \in H$ that is consistent with S. The consistency problem is defined as follows:

Definition 2.2 (Consistency problem $Con(C,H)$ [15]).
Instance *A labeled sample S of some $c \in C$.*
Solution *$h \in H$ such that h is consistent with S and $|h|$ is minimized.*

We denote by $Con(C,H)$ the problem of finding a minimal $h \in H$ that is consistent with some labeled sample S of some $c \in C$ and likewise we call such a minimal consistent h a solution to the consistency problem of an instance S of $Con(C,H)$. Occam's razor makes a connection between the consistency problem and the ability to learn one representation class by another. In this context learning is defined as follows: Let $0 \leq \epsilon < 1$ and $0 < \delta \leq 1$. An (ϵ, δ)-*probably approximately correct* (PAC) [30] learning algorithm for C by H outputs an $h \in H$, such that $\text{error}_{c,D}(h) \leq \epsilon$ with probability at least $1 - \delta$ (for all distributions D over X and all $c \in C$).

We are now in the position to introduce the core theorem of this section, which connects the task of PAC learning and an approximation task. Intuitively, the following theorem states that finding a hypothesis that explains the observed data (i.e., is consistent with S) and is significantly more compact than the data, is sufficient for PAC learning.

Theorem 2.3 (Occam's razor [5, 15])**.** *Given a labeled sample S of c of size*

$$m = O\left(\frac{1}{\epsilon}\log\frac{1}{\delta} + \left(\frac{n^\alpha}{\epsilon}\log\frac{n^\alpha}{\epsilon}\right)^{1/(1-\beta)}\right), \tag{8}$$

where the m examples have been sampled independently from D and for some fixed $\alpha \geq 1$ and $0 \leq \beta < 1$, any h that is consistent with S and which satisfies

$$|h| \leq opt_{Con}(S)^\alpha \, |S|^\beta \tag{9}$$

does also satisfy $\mathrm{error}_{c,D}(h) \leq \epsilon$ *with probability at least $1-\delta$.*

Here, α and β are fixed values for the Occam's razor prescription, the intuition for them being hinted at in Ref. [5]. When m is fixed to a sufficiently large number, fulfilling the scaling of the above theorem, then α can be seen as reflecting the property that $opt_{Con}(S)^\alpha$ bounds some polynomial in $opt_{Con}(S)$ and β can hence be viewed as a "compression parameter". If $\beta = 0$, we have complete compression. Then the algorithm provides a consistent hypothesis of complexity at most $opt_{Con}(S)^\alpha$, independent of the sample size. The sample size needed is then $m = O(\frac{1}{\epsilon}\log\frac{1}{\delta})$. For $\beta \to 1$, we actually have not learned much, since almost all of S can be encoded in h.

Then, note that the size of S is a polynomial in $(n, \frac{1}{\epsilon}, \frac{1}{\delta})$. The variable α resembles that $|h|$ must be smaller than some polynomial in the optimal solution size, while β forces that h does not simply hard-encode S. Clearly, it follows that any algorithm that for all $c \in C$ and all D, on input S sampled according to D of size $\mathrm{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$, outputs an $h \in H$ with $|h|$ upper bounded as in Theorem 2.3 is a PAC learning algorithm for C by H. Importantly, learning C by H can be interpreted as an approximation task. Specifically, the task is to approximate the optimal solution $opt_{Con}(S)$, which is the size of the smallest representation consistent with S, by $|h|$, where h is a representation that is also consistent with S, for any S of sufficient size. An algorithm achieving such an approximation within a factor of $opt_{Con}(S)^{\alpha-1}|S|^\beta$, for all S with $|S| = \mathrm{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$, is an (ϵ, δ)-PAC learner for C. In the remainder of this work, when we say that some "algorithm approximates the solution of the $Con(C, H)$ problem", we mean that the algorithm outputs an h, such that $|h|$ approximates $opt_{Con}(S)$ by a factor $opt_{Con}(S)^{\alpha-1}|S|^\beta$, where h has the important property of being consistent with S. This sense of approximation might seem unnatural, but the Con problem will later be reduced to a combinatorial optimization task, where it is natural to approximate some scalar quantity and satisfy some constraints.

2.6 Formula coloring problem

We now introduce the *formula coloring problem* (FC) that takes the centre stage in our later argument. It is a combinatorial optimization problem that has originally been introduced in Ref. [15] as a generalization of the more common *graph coloring problem.* It is an optimization problem of the type as is frequently considered in notions of quantum approximate optimization: In fact, in a subsequent section, we will formulate this problem as a problem of minimizing the energy of a commuting local Hamiltonian, to make that connection explicit. It is one of the main results of this work to show a super-polynomial quantum advantage for FC and integer programming. Let $z_1, \ldots, z_m \in \mathbb{N}$ be the *variables* in a Boolean formula, each being assigned an integer value, which acts as the integer valued *color* of the variable. That is to say, each of the variables $z_1, \ldots, z_m$ takes exactly one of the possible values referred to as colors. We regard an assignment of colors to the z_i (called a coloring) as a partition of the variable set into equivalence classes. That is to say, two variables have the same color if and only if they are in the same equivalence class. For the FC problem, we consider Boolean formulae $F(z_1, \ldots, z_m)$ which consist of conjunctions of two types of clauses. On the one hand, these are clauses of the form $(z_i \neq z_j)$. This is, in fact, precisely of the form as the clauses of the more common *graph coloring problem.* On the other hand, there are clauses of the form $((z_i = z_j) \to (z_k = z_l))$. This material conditional, as it is called in Boolean logic, can equivalently and possibly more commonly be written as

$$((z_i \neq z_j) \vee (z_k = z_l)). \tag{10}$$

A *coloring* is an assignment of colors to the z_i, described by a partitioning P of the variable set into k equivalence classes, i.e., $|P| = k$. This means that $z_i = z_j$ if and only if they are in the same partition of the k partitions in P. We are now in the position to formulate the formula coloring problem.

Definition 2.4 (Formula coloring problem FC [15])**.**
Instance *A Boolean formula $F(z_1, \ldots, z_m)$ which consists of conjunctions of clauses of the form either $(z_i \neq z_j)$ or the form $((z_i = z_j) \to (z_k = z_l))$.*
Solution *A minimal coloring P for $F(z_1, \ldots, z_m)$ such that F is satisfied.*

A *minimum solution* to the FC problem is a coloring with the fewest colors, i.e., $|P|$ is minimal for all possible colorings such that F is satisfied. The example given in Ref. [15] is the formula

$$(y_1 = y_2) \vee ((y_1 \neq y_2) \wedge (y_3 \neq y_4)) \tag{11}$$

has as a model the two-color partition $\{y_1, y_3\}$, $\{y_2, y_4\}$ and has as a minimum model the one-color partition $\{y_1, y_2, y_3, y_4\}$. The formula coloring problem is obviously NP-complete, as the problem is in NP and graph coloring is NP-hard.

2.7 The RSA encryption function

Throughout this work, we will make use on the hardness of inverting the RSA encryption function [24], which forms the foundation of the security of the RSA public-key cryptosystem, one of the canonical public-key crypto-systems and presumed to be secure against classical adversaries [10].

Let $N = p \times q$ be the product of two primes p and q, both of similar bit-length. Define *Euler's totient function* ϕ, where $\phi(N)$ is equal to the number of positive integers up to N that are relative prime to N. It holds that $x^{\phi(N)} = 1 \bmod N$. When two parties, which we refer to as Bob and Alice, wish to communicate via an authenticated but public channel, they can do so as follows: First, Alice generates two primes p and q of similar bit-length and computes their product $N = p \times q$. Then, Alice generates a so-called public-private key pair (d, e), where d is the *secret key* satisfying $d \times e \bmod \phi(N) = 1$, and e is the *public exponent.* Alice shares the public key (e, N) with Bob over the public channel. We define the RSA encryption function for a given exponent e, a message $x \in \mathbb{Z}_N$, and a *modulus* N as

$$RSA(x, N, e) = x^e \bmod N. \tag{12}$$

To encrypt a message $x \in \mathbb{Z}_N$, Bob simply computes the output of the RSA encryption function, given N and e. Bob then sends the ciphertext $c = x^e \bmod N$ to Alice, who decrypts the ciphertext by computing $c^d \bmod N = (x^e)^d \bmod N = x^{1+i\times\phi(N)} \bmod N = x \bmod N$, where the last step follows from the fact that $x^{\phi(N)} = 1 \bmod N$ and $e \times d = 1 + i \times \phi(N)$ for some $i \in \mathbb{N}$ because $e \times d \bmod \phi(N) = 1$.

The security of the RSA cryptosystem is closely related on the presumed hardness of integer factoring and, more generally, is based on the presumed hardness of *inverting* the RSA encryption function without knowledge of the secret key d. That is, there is no known classical polynomial-time algorithm that, given $(\mathrm{RSA}(x, N, e), N, e)$ outputs x. On a quantum computer, however, Shor's algorithm [25] can be used to factor the integer N in polynomial time. This immediately gives rise to a quantum polynomial time algorithm that inverts the RSA encryption function; Simply factor the public modulus using Shor's algorithm, and then compute $\phi(N) = (p - 1) \times (q - 1)$. Then, one can find a d such that $e \times d \bmod \phi(N) = 1$ by using the *extended Euclidean algorithm.* In summary, under the standard cryptographic assumption that the RSA encryption function is hard to invert, Shor's algorithm thus gives rise to a computational quantum-classical separation. As we will show, this separation extends to the approximation of combinatorial optimization problems as well.

Throughout this work, we will make use of the fact that determining the *least significant bit* (LSB) of x, given $\mathrm{RSA}(x, N, e)$ is as hard as inverting the RSA encryption function. Formally, Alexi et. al. [1] have proven that if there exists a classical polynomial-time algorithm that finds the LSB of x, given $RSA(x, N, e)$, then there exists a classical polynomial-time algorithm that inverts the RSA encryption function.

3 Classical hardness of approximation

To show our quantum advantage, we require a classical hardness result and quantum efficiency result. In this section, we establish the classical hardness of approximating combinatorial optimization solutions. We build on the results of Ref. [15], where the hardness of approximation tasks has been established. Furthermore, their work shows how the these hard-to-approximate problems can be reduced to the combinatorial optimization problem of *formula coloring*. We then extend these results by showing an approximation-preserving reduction from formula coloring to *integer linear programming* (ILP). These results will constitute the classical hardness part for the quantum-classical separation we show. Figure 3 gives a high level overview of the results presented

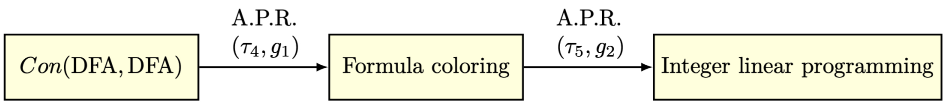


Figure 3: In Section 3.1, we introduce Boolean circuits, whose sizes are hard to approximate by $|h|$, where h is a hypothesis that is consistent with a sample labeled by the circuits. This directly implies the approximation hardness of $Con(\mathrm{DFA}, \mathrm{DFA})$. In Section 3.2, we present an approximation-preserving reduction from $Con(\mathrm{DFA}, \mathrm{DFA})$ to formula coloring [15]. We then extend the results of Ref. [15] by showing in Section 3.3 an approximation-preserving reduction from formula coloring to integer linear programming, yielding the approximation hardness for ILP.

in this section.

3.1 Approximation hardness of the Con problem

In this subsection, we present the result that approximating the solution of $Con(\mathrm{DFA}, \mathrm{DFA})$ is hard using a classical computer [15]. This result is obtained through the assumption that inverting the RSA encryption function is hard, a widely accepted cryptographic assumption. To do this, one defines a class of Boolean circuits that essentially decrypt a given RSA ciphertext and output the LSB of the cleartext. Intuitively, the authors of Ref. [15] show that, since PAC learning these Boolean circuits is hard (otherwise one would be able to invert RSA), the approximation of these decryption circuits by any polynomially evaluatable representation class in the sense of Theorem 2.3 must also be hard, using a classical computer. They then show that this implies that approximating the solution of $Con(\mathrm{DFA}, \mathrm{DFA})$ must also be hard. To follow the argumentation in Ref. [15], let $N \in \mathbb{N}$ and $x \in \mathbb{Z}_N$ and define

$$\mathrm{powers}_N(x) := x \bmod N, x^2 \bmod N, x^4 \bmod N, \ldots, x^{2^{\lceil log(N) \rceil}} \bmod N \tag{13}$$

as the sequence of the first $\lceil \log(N) \rceil + 1$ square powers of x.

Definition 3.1 (Boolean circuit for the LSB of RSA [15])**.** *Let* $\text{C-RSA}_n \subset \mathrm{BC}_n$ *and* $\text{C-RSA} = \bigcup_{n \geq 1} \text{C-RSA}_n$ *be the representation class of* log-depth, poly-size Boolean circuits *that, on input* binary $(\mathrm{powers}_N(\mathrm{RSA}(x, N, e)), N, e)$*, output $LSB(x)$ for all $x \in \mathbb{Z}_N$. Each representation in* C-RSA_n *is defined by a triple (p, q, e) and this representation will be denoted $r_{(p,q,e)}$, where p and q are primes of exactly $n/2$ bits and $e \in \mathbb{Z}_N$ and $N = p \cdot q$.*
An example of $r_{(p,q,e)} \in$ C-RSA_n *is of the form*

$$\left(\text{binary}\left(\mathrm{powers}_N(\mathrm{RSA}(x, N, e)), N, e\right), LSB(x)\right), \textit{ with } x \in \mathbb{Z}_N. \tag{14}$$

It is important to note at this point that the calculation of the LSB of x, given the input binary $(\mathrm{powers}_N(\mathrm{RSA}(x, N, e)), N, e)$ can indeed be performed by a $O(\log(n))$-depth, poly(n)-size Boolean circuit, if the decryption key d is known [15]. In Fig. 4, we depict a schematic picture of such a Boolean circuit in C-RSA. Since learning the LSB of the cleartext is as hard as inverting the RSA function [1], which is widely assumed to be intractable for classical computers, Kearns and Valiant [15] show the classical approximation hardness of $Con(\text{C-RSA}, H)$, where H is any polynomially evaluatable representation class. The following theorem states that (assuming the classical hardness of inverting RSA) and given some sample S of some $r_{(p,q,e)} \in$ C-RSA, no

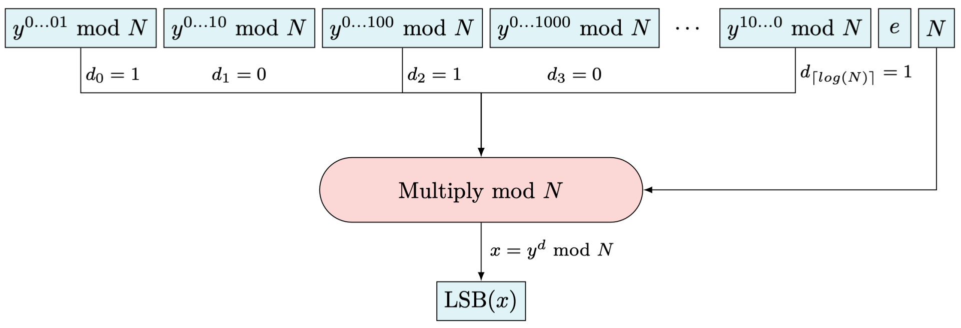


Figure 4: The input to the circuit in C-RSA is the power sequence of the RSA ciphertext of $\mathrm{RSA}(x, N, e) = y$. The circuit computes the LSB of x by simply performing modular multiplication on the 2^i'th powers of the power sequence where the secret key bit $d_i = 1$, for the secret key d. Thereby the secret key d is hard-wired into the circuit and the decryption $x = y^d$ mod N is explicitly performed. This can be done in an $O(\log(n))$ deep circuit [4].

polynomial-time classical algorithm can output a hypothesis $h \in H$ that is consistent with S and only polynomially larger than the smallest possible hypothesis.

Theorem 3.2 (Classical approximation hardness of C-RSA [15])**.** *Let H be any polynomially evaluatable representation class. Assuming the hardness of inverting the RSA function, there exists no classical probabilistic polynomial-time algorithm that on input an instance S of $Con(\text{C-RSA}, H)$ finds a solution $h \in H$ that is consistent with S and approximates the size $opt_{Con}(S)$ of the optimal solution by*

$$|h| \leq (opt_{Con}(S))^{\alpha}|S|^{\beta}$$

for all S and any $\alpha \geq 1$ and $0 \leq \beta < 1$.

Since $|S| = n \times m = n \times \mathrm{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ and $\alpha \geq 1$ we get that the optimal size $opt_{Con}(S)$ cannot be approximated up to a polynomial factor, holding for all classical probabilistic polynomial-time algorithms, where the sense of approximation is explained in detail in Section 2.5.

3.1.1 Classical approximation hardness for more representation classes

Furthermore, Kearns and Valiant [15] show that the approximation hardness of C-RSA implies approximation hardness for *Boolean formulae*, *log-space Turing machines* and DFAs. In particular, let BF-RSA be the class of *Boolean formulae* that we obtain when we reduce every instance in C-RSA using τ_1, i.e., BF-RSA $= \{F \mid (F, x) = \tau_1(I) \text{ and } I \text{ instance of } Eval(\text{C-RSA})\}$. In a similar manner, LSTM-RSA is the class of *log-space Turing machines* that we obtain when we reduce BF-RSA using τ_2 and finally, DFA-RSA is the class of DFAs that we obtain when using τ_3 on BF-RSA. Since the evaluation problem of resulting representations are poly-time reducible to each other and are at most polynomially larger, the following holds [15]:

Theorem 3.3 (Classical approximation hardness of more representations [15])**.** *Let H be any polynomially evaluatable representation class. Assuming the hardness of inverting the RSA function, there exists no classical probabilistic polynomial-time algorithm that, on input an instance S of (a) $Con(\text{BF-RSA}, H)$, (b) $Con(\text{LSTM-RSA}, H)$, or (c) $Con(\text{DFA-RSA}, H)$, finds a solution $h \in H$ that is consistent with S and approximates the size $opt_{Con}(S)$ of the optimal solution by*

$$|h| \leq (opt_{Con}(S))^{\alpha}|S|^{\beta}$$

for all S and any $\alpha \geq 1$ and $0 \leq \beta < 1$.

Specifically, note that approximating the solution of $Con(\text{DFA-RSA}, \text{DFA})$ is at least as hard as to approximate the solution of $Con(\text{DFA-RSA}, H)$.

3.2 Approximation hardness of formula coloring

In this work we are interested in showing a quantum advantage for approximating the solution of combinatorial optimization problems. Therefor, we require a classical approximation hardness result for a combinatorial optimization problem. To that end, the work of Kearns and Valiant [15] gives an approximation-preserving reduction from the $Con(\text{DFA}, \text{DFA})$ problem to the *formula coloring problem*, which is a combinatorial optimization problem. We denote the approximation-preserving reduction from $Con(\text{DFA}, \text{DFA})$ to FC by (τ_4, g_1), where we will explicitly give the construction of the instance transformation τ_4, which maps an instance S of $Con(\text{DFA}, \text{DFA})$ to an instance F_S of FC. First, observe that S contains the examples $(w_1, b_1), (w_2, b_2), \ldots, (w_m, b_m)$ where $w_i \in \{0,1\}^k$ and the labels $b_i \in \{0,1\}$. The formula F_S will be over variables z_i^j, where $1 \leq i \leq m$ and $0 \leq j < k$. Essentially, each variable z_i^j will correspond to the state that a consistent DFA would be in after reading the j-th bit of w_i.

We now give the construction for the formula F_S: For each i_1, i_2 and j_1, j_2, such that $0 \leq j_1, j_2 < k$ and $w_{i_1}^{j_1+1} = w_{i_2}^{j_2+1}$, we add the predicate

$$((z_{i_1}^{j_1} = z_{i_2}^{j_2}) \rightarrow (z_{i_1}^{j_1+1} = z_{i_2}^{j_2+1})) \tag{15}$$

to the conjunctions in F_S. Intuitively, this encodes that for two inputs w_{i_1}, w_{i_2}, a DFA that is in the same state for both inputs and then reads the same symbol for both those strings next, the resulting state should also be the same. To ensure the DFA is consistent with the labels of the sample as well, for each $1 \leq i_1, i_2 \leq m$, such that $b_{i_1} \neq b_{i_2}$, we add the predicate

$$(z_{i_1}^k \neq z_{i_2}^k) \tag{16}$$

to the conjunctions in F_S. Those clauses encode the fact that for different labels, the states (after reading the whole input) of a consistent DFA must be different, since any state can either only accept or reject.

If $|S|$ is the number of bits in S, the resulting F_S consists of $\Theta(|S|^2)$ many clauses. For the solution transformation g_1, as well as the proof that this reduction is indeed correct we refer to the proof in Ref. [15]. It is important to note that by the construction above, the bits of the examples are now encoded in the clauses of F_S together with the correct working of the DFA and the solution (the structure of the minimal DFA) is the minimal coloring of F_S. Due to the results of Ref. [15], the following theorem holds:

Theorem 3.4 (Reduction of $Con(\text{DFA}, \text{DFA})$ to FC [15])**.** *There is a polynomial time algorithm τ_4 that on input an instance S of the problem $Con(\text{DFA}, \text{DFA})$ outputs an instance F_S of the* formula coloring problem *such that S has a k-state consistent hypothesis $M \in$ DFA if and only if F_S has a coloring P, with $|P| = k$.*

Note that the algorithm τ_4 is precisely the instance transformation of the reduction (τ_4, g_1) and we have:

$$Con(\text{DFA}, \text{DFA}) \leq_p \text{FC}\,. \tag{17}$$

In particular, it holds that

$$Con(\text{DFA-RSA}, \text{DFA}) \leq_p \text{FC-RSA}, \tag{18}$$

where FC-RSA is the class of *formula coloring problems* that result out of running τ_4 on the instances in the problem $Con(\text{DFA-RSA}, \text{DFA})$. In particular, g_1 transforms the minimal solution of FC into the minimal solution of $Con(\text{DFA}, \text{DFA})$, thus $opt_{\text{FC}}(F_S) = opt_{Con}(S)$ (due to Theorem 3.4) and $|F_S| = \Theta(|S|^2)$. From those two facts, it follows that finding a valid coloring P of F_S, such that $|P| \leq opt_{FC}(F)^\alpha |F|^{\beta'}$ would contradict Theorem 3.3, for the parameter range $\alpha \geq 1$, $0 \leq \beta' < 1/2$. Thus, the reduction (τ_4, g_1) preserves the approximation hardness of $Con(\text{DFA-RSA}, \text{DFA})$ in the sense of the following theorem [15]:

Theorem 3.5 (Classical hardness of approximation for *formula coloring* [15])**.** *Assuming the hardness of inverting the RSA function, there exists no classical probabilistic polynomial-time algorithm* *that on input an instance F_S of* FC-RSA *finds a valid coloring P that approximates the size $opt_{\mathrm{FC}}(F_S)$ of the optimal solution by*

$$|P| \leq opt_{\mathrm{FC}}(F)^{\alpha}|F|^{\beta} \tag{19}$$

for any $\alpha \geq 1$ and $0 \leq \beta < 1/2$.

In a similar mindset, we present an approximation preserving reduction of FC to the *integer linear programming problem* in the subsequent section.

3.3 Approximation hardness of integer linear programming

In this section, we show an approximation-preserving reduction of the *formula coloring problem* to the problem of *integer linear programming.* ILP is an NP-complete problem in which many practically relevant combinatorial optimization tasks are formulated, such as planning or scheduling tasks [32]. The problem is to minimize (or maximize) an objective function that depends on integer variables. Additionally, there are constraints on the variables that need to be followed. Let us define an ILP problem within our formalism:

Definition 3.6 (Integer linear programming problem (ILP))**.**
Instance *An objective function over integer variables subject to constraints of the variables.*
Solution *A valid assignment of the variables under the constraints, such that the objective function is minimal.*

We now show the reduction (τ_5, g_2) of *formula coloring* to ILP, by first giving the instance transformation τ_5:

Let $F(z_1, \ldots, z_M)$ be a formula coloring instance over variables $z_1, \ldots, z_M \in \mathbb{N}$ which is a conjunction of Q clauses of the form $(z_u \neq z_v)$ and R clauses of the form $((z_u = z_v) \rightarrow (z_k = z_l))$ (which is equivalent to $((z_u = z_v) \vee (z_k \neq z_l))$). For $1 \leq u, i \leq M$ and $1 \leq j \leq R$ we introduce the ILP variables $w_i, x_{u,i}, a_j, b_j, s_j \in \{0, 1\}$ and $1 \leq \hat{z}_u \leq M$, where $\hat{z}_u$ resembles the variable z_u in F and w_i indicates if the i'th color is used and $x_{u,i}$ indicates if the variable $\hat{z}_u = i$ and a_j, b_j, s_j are helper variables.

It is important to note that for some k-coloring $P = \{P_1, \ldots, P_k\}$ of F, the clause $(z_u = z_v)$ in F is true iff $z_u, z_v \in P_i$ for some color i. On the other hand, the clause $(z_u \neq z_v)$ in F is true iff $z_u \in P_i$ and $z_v \notin P_i$ for some color i. In our ILP construction, we introduce an analogue variable to z_u, namely $\hat{z}_u$, where $\hat{z}_u$ directly takes as value the color i, i.e., $\hat{z}_u = i$ iff $z_u \in P_i$. By our construction, we get the *integer linear programming* problem ILP_F

$$\text{minimize} \sum_{1 \leq i \leq M} w_i \tag{20}$$

subject to the following constraints,

$$\begin{aligned}
&\text{for all } u, i \in \{1, \ldots, M\}, & (x_{u,i} = 1) \iff (\hat{z}_u = i), && (21)\\
&\text{for all } u \in \{1, \ldots, M\}, & \sum_{i=1}^{M} x_{u,i} = 1, && (22)\\
&\text{for all } u, i \in \{1, \ldots, M\}, & x_{u,i} \leq w_i, && (23)\\
&\text{for all } Q \text{ clauses } (z_u \neq z_v) \text{ and all } i \in \{1, \ldots, M\}, & x_{u,i} + x_{v,i} \leq 1, && (24)\\
&\text{for all } R \text{ clauses } ((z_u \neq z_v) \vee (z_k = z_l)) \text{ with } j \in \{1, \ldots, R\}, & (a_j = 1) \iff (\hat{z}_k = \hat{z}_l), && (25)\\
& & (b_j = 1) \iff (\hat{z}_u \neq \hat{z}_v), && (26)\\
& & s_j = (a_j \vee b_j), && (27)\\
& & s_j \geq 1, && (28)
\end{aligned}$$

$$\text{and } w_i, x_{u,i}, a_j, b_j, s_j \in \{0, 1\} \text{ and } 1 \leq \hat{z}_u, \hat{z}_v, \hat{z}_k, \hat{z}_l \leq M. \tag{29}$$

Before explaining the constraints, let us note that for the sake of understanding, we display here logical clauses in (21), (25), (26) and (27), even though they are technically not ILP constraints.

We refer to Appendices A.1 and A.2 on how the logical clauses in (21), (25), (26) and (27) are concretely converted to inequality constraints.

We define the binary variable w_i to be 1 iff color i is used. Hence, the minimization task at hand over the $w_i's$ corresponds to finding the minimal coloring of F. Constraint (21) defines the binary variable $x_{u,i}$ to be 1 iff $\hat{z}_u = i$, i.e., indicating that $z_u \in P_i$. Constraint (22) ensures that any variable is assigned to exactly one color. Constraint (23) ensures that if there is some $z_u \in P_i$, then $w_i = 1$, since color i is used. Constraint (24) encodes the $(z_u \neq z_v)$ clauses in F, i.e., that z_u, z_v are not assigned the same color. Constraints (25), (26), (27) and (28) encode the $((z_u \neq z_v) \vee (z_k = z_l))$ clauses in F.

In total, we get $M(4M + Q + 1) + 12R$ constraints and $2M(M + 1) + 5R$ variables, which are polynomial in the size of F. Thus τ_5 is indeed computable in polynomial time. Now the solution transformation g_2 simply works by partitioning the variables z_u, z_v into the same set iff $\hat{z}_u = \hat{z}_v$. Clearly, g_2 is computable in polynomial time. We show that (τ_5, g) is indeed a reduction of FC to ILP by proving an even stronger result:

Theorem 3.7 (Reduction of FC to ILP)**.** *Let τ_5 be a polynomial-time algorithm that on input an instance $F(z_1, \ldots, z_M)$ of the* formula coloring problem FC *outputs an instance* ILP$_F$ *of the* integer linear programming *problem. Let g_2 be a polynomial-time algorithm that on input an assignment A of* ILP$_F$ *outputs a coloring P of F. There exist τ_5, g_2, such that P is a valid k-coloring of F if and only if A is a valid assignment of the variables in* ILP$_F$ *such that the objective function of* ILP$_F$ *is k.*

Proof. Let τ_5 and g_2 be the algorithms described in the beginning of this section.
$\underline{\Longrightarrow:}$ We first prove that if F has a valid coloring P of k colors, then there exists an assignment A of the variables such that

$$\sum_{1 \le i \le M} w_i = k. \tag{30}$$

Without loss of generality, assume an ordering of the sets in $P = \{P_1, \ldots, P_k\}$. Since P is a coloring of F, the P_i's are pairwise disjoint. We assign the variables in ILP$_F$ as follows.

For all $i \in \{1, \ldots, M\}$, $$w_i = \mathbb{1}(i \le k), \tag{31}$$

for all $u \in \{1, \ldots, M\}$, $$\hat{z}_u = \sum_{i=1}^{M} \mathbb{1}(z_u \in P_i) \times i, \tag{32}$$

for all $u, i \in \{1, \ldots, M\}$, $$x_{u,i} = \mathbb{1}(z_u \in P_i), \tag{33}$$

for all R clauses $((z_u \neq z_v) \vee (z_k = z_l))$ with $j \in \{1, \ldots, R\}$, $$a_j = \mathbb{1}(\hat{z}_k = \hat{z}_l), \tag{34}$$

$$b_j = \mathbb{1}(\hat{z}_u \neq \hat{z}_v), \tag{35}$$

$$s_j = a_j + b_j. \tag{36}$$

Clearly, the objective function of ILP$_F$ is k. It remains to be shown that the constraints in ILP$_F$ are satisfied. First, note that from the variable assignments it follows that $(\hat{z}_u = i) \iff (\mathbb{1}(z_u \in P_i) = 1)$. We can then see that the constraint (21) is satisfied, since

$$(x_{u,i} = 1) \iff (\mathbb{1}(z_u \in P_i) = 1) \iff (\hat{z}_u = i). \tag{37}$$

The constraint (22) is satisfied, due to the pairwise disjointedness of the sets in P and we get

$$\sum_{i=1}^{M} x_{u,i} = \sum_{i=1}^{M} \mathbb{1}(z_u \in P_i) = 1. \tag{38}$$

Next, we turn our attention to constraint (23). To see why this constraint is satisfied observe the following. From the fact that $\sum_{i=1}^{M} x_{u,i} = 1$ it follows that there is exactly one i', for which $x_{u,i'} = 1$. By the definition of $x_{u,i'}$, we have $\mathbb{1}(z_u \in P_{i'}) = 1$. Since $P = \{P_1, \ldots, P_k\}$ and $P_{i'}$ is not empty, it must hold that $i' \le k$ and hence by construction $w_{i'} = 1$. For all other $i \neq i'$, we have $x_{u,i} = 0$, and thus $x_{u,i} \le w_i$ and constraint (23) is satisfied. The constraint (24) is satisfied, since we have

$$x_{u,i} + x_{vi} = \mathbb{1}(z_u \in P_i) + \mathbb{1}(z_v \in P_i) \le 1 \tag{39}$$

because of the assumption that P is a valid coloring and this constraint occurs only for clauses of the form $(z_u \neq z_v)$. The constraints (25) and (26) are satisfied by definition. One can easily see that (27) and (28) are also satisfied, since P is a valid coloring and these constraints only occur for clauses of the form $((z_u \neq z_v) \vee (z_k = z_l))$.

$\underline{\Longleftarrow:}$ Assume that we are given a valid assignment A of the variables in ILP_F, such that $\sum_{1\leq i\leq M} w_i = k$. Then, we can construct a valid coloring P for the corresponding formula coloring instance F. To this end, run g_2 by partitioning the variables z_u, z_v into the same sets iff $\hat{z}_u = \hat{z}_v$. Since

$$\sum_{1\leq i\leq M} w_i = k, \tag{40}$$

there exist $i_1, \ldots, i_k$ for which $w_{i_1}, \ldots, w_{i_k} = 1$. Since for all $u \in \{1, \ldots, M\}$ we have

$$\sum_{i=1}^{M} x_{u,i} = 1 \tag{41}$$

and $x_{u,i} \leq w_i$, there exist $u_1, \ldots, u_k$ for which $x_{u_1,i_1}, \ldots, x_{u_k,i_k} = 1$. Therefore, since the $u_1, \ldots, u_k$ are pairwise different and $i_1, \ldots, i_k$ are pairwise different and because $(x_{u,i} = 1) \iff (\hat{z}_u = i)$, there are $\hat{z}_{u_1} = i_1, \ldots, \hat{z}_{u_k} = i_k$ that are different from each other. Therefore, if we partition variables z_u, z_v into the same partition iff $\hat{z}_u = \hat{z}_v$, we obtain exactly k partitions. Now we need to show that this coloring is a valid coloring for F. The clauses $(z_u \neq z_v)$ are satisfied since constraints (24) and (21) are satisfied. The clauses $((z_u \neq z_v) \vee (z_k = z_l))$ are satisfied since constraints (25), (26), (27), (28) are satisfied. This ends the proof of the reduction. □

Thus, we have that

$$\mathrm{FC} \leq_p \mathrm{ILP} \tag{42}$$

and in particular

$$\text{FC-RSA} \leq_p \text{ILP-RSA}, \tag{43}$$

where ILP-RSA are the instances of ILP that we get when we apply τ_5 to all instances of FC-RSA. Since, by the same arguments as in Section 3.2 and since g_2 transforms the minimal solution of ILP to the minimal solution of FC and $|\,\mathrm{ILP}_F\,| = \Theta(|F|^2)$, the reduction (τ_5, g_2) preserves the approximation hardness of FC-RSA, in the sense of the following theorem.

Theorem 3.8 (Classical hardness of approximation for *integer linear programming*)**.** *Assuming the hardness of inverting the RSA function, there exists no classical probabilistic polynomial-time algorithm that on input an instance* ILP_F *of* FC-RSA *finds an assignment of the variables in* ILP_F *which satisfies all constraints and approximates the size* $opt_{\mathrm{ILP}}(\mathrm{ILP}_F)$ *of the optimal solution by*

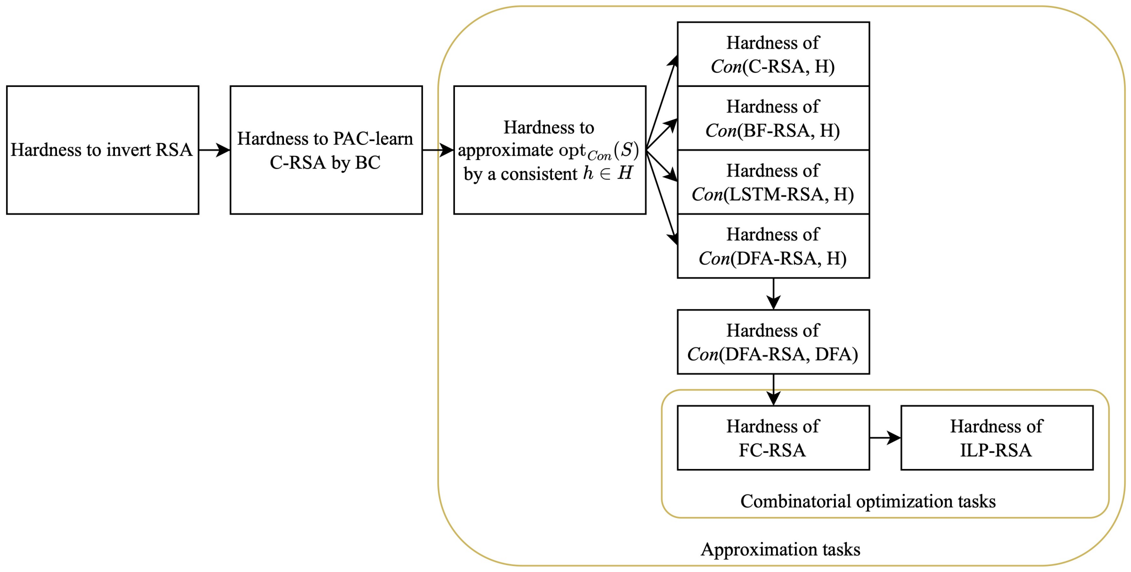
$$\sum_{1\leq i\leq M} w_i \leq opt_{\mathrm{ILP}}(\mathrm{ILP}_F)^{\alpha} |\,\mathrm{ILP}_F\,|^{\beta} \tag{44}$$

for any $\alpha \geq 1$ *and* $0 \leq \beta < 1/4$.

To give a high-level overview of the hardness results established in this section, we present in Fig. 5 the chain of implications.

4 Quantum efficiency

In the previous section we presented proofs for the classical hardness of various approximation tasks. In this section, we show a quantum advantage by proving that the instances resulting from the reductions described in Section 3 can be solved in polynomial time given access to a fault-tolerant quantum computer. This yields the desired result of *quantum separation* for natural problems: Under the assumption that inverting the RSA function is hard, quantum computers can find close to optimal solutions to problem instances for which classical computers are incapable of findings solutions of the same quality.


Hardness to invert RSA
Hardness to PAC-learn C-RSA by BC
Hardness to approximate $\text{opt}_{Con}(S)$ by a consistent $h \in H$
Hardness of *Con*(C-RSA, H)
Hardness of *Con*(BF-RSA, H)
Hardness of *Con*(LSTM-RSA, H)
Hardness of *Con*(DFA-RSA, H)
Hardness of *Con*(DFA-RSA, DFA)
Hardness of FC-RSA
Hardness of ILP-RSA
Combinatorial optimization tasks
Approximation tasks
Figure 5: The argument chain that propagates the hardness to invert the RSA function to the hardness of approximating combinatorial optimization tasks.

First, we demonstrate that the solutions to instances of $Con(\text{C-RSA}, \text{BC})$ can be approximated by a polynomial factor in quantum polynomial time leveraging Shor's algorithm. Later, we show approximation separation results for more "natural" problems, namely *formula coloring* and *integer linear programming*.

Theorem 4.1 (Quantum efficiency for approximating the solution of $Con(\text{C-RSA}, \text{BC})$)**.** *There exists a polynomial-time quantum algorithm that, on input of an instance S of $Con(\text{C-RSA}, \text{BC})$, finds a consistent hypothesis $h \in \text{BC}$ which approximates the size $opt_{Con}(S)$ of the optimal solution by*

$$|h| \leq opt_{Con}(S)^{\alpha} \tag{45}$$

for all S and for some $\alpha \geq 1$.

Proof. Let S be an instance of $Con(\text{C-RSA}, \text{BC})$. In Algorithm 1, on input S, a hypothesis circuit

Algorithm 1: Approximate the solution of $Con(\text{C-RSA}, \text{BC})$

```
Input  : A labeled sample $S$ of C-RSA
Output : The description of a Boolean circuit consistent with $S$
Pick any example $s \in S$ and read $e, N$ from it;
Run Shor's algorithm [25] to factor $N$ and retrieve $p$ and $q$;
Run the extended Euclidean algorithm to compute $d$, such that
  $d \times e = 1 \bmod (p-1)(q-1)$;
// Note that at this point, $d$ is the secret RSA exponent.
Output the description of a Boolean circuit that, on input
  binary $(\text{powers}_N(\text{RSA}(x, N, e)), N, e)$, multiplies the $2^i$'th powers together for which the
  bit $d_i = 1$ (thereby hard-wiring $d$ into the circuit), using the iterated products technique
  [4] and outputs the LSB of the result.
```

h is output, which is of size $\text{poly}(n)$ and which explicitly decrypts a RSA ciphertext given by its power series. We know from Section 3.1 that h is consistent with S and of polynomial size. It is clearly the case that $n \leq opt_{Con}(S)$ and thus it holds that

$$|h| \leq n^{\alpha} \leq opt_{Con}(S)^{\alpha}. \tag{46}$$

□

Contrasted with the explicit approximation hardness from Theorem 3.2, this yields the super-polynomial advantage of quantum algorithms over classical algorithms for the specific approximation task, namely approximating the optimal consistent hypothesis size by $|h|$ with h consistent with S. We can indeed obtain similar results also for $Con(\mathrm{BF\text{-}RSA}, \mathrm{BF})$, $Con(\mathrm{LSTM\text{-}RSA}, \mathrm{LSTM})$ and $Con(\mathrm{DFA\text{-}RSA}, \mathrm{DFA})$. In particular, given S, we can use Algorithm 1 to obtain a consistent h of C-RSA and then leverage the poly-time instance transformations τ_1, τ_2, τ_3, to obtain an at most poly(n) larger approximation to the solution of $Con(\mathrm{BF\text{-}RSA}, \mathrm{BF})$, $Con(\mathrm{LSTM\text{-}RSA}, \mathrm{LSTM})$ and $Con(\mathrm{DFA\text{-}RSA}, \mathrm{DFA})$. Thus, we obtain the following corollary:

Corollary 4.2 (Quantum efficiency for more approximation tasks)**.** *There exists a polynomial-time quantum algorithm that, on input an instance S of (a) $Con(\mathrm{BF\text{-}RSA}, \mathrm{BF})$, (b) $Con(\mathrm{LSTM\text{-}RSA}, \mathrm{LSTM})$, or (c) $Con(\mathrm{DFA\text{-}RSA}, \mathrm{DFA})$, finds a consistent hypothesis (a) $h \in \mathrm{BF}$, (b) $h \in \mathrm{LSTM}$, (c) $h \in \mathrm{DFA}$ which approximates the size $opt_{Con}(S)$ of the optimal solution by*

$$|h| \leq opt_{Con}(S)^{\alpha}$$

for all S and for some $\alpha \geq 1$.

This again yields super-polynomial advantages of quantum algorithms over classical algorithms for approximating the optimal solution size of the consistency problem by the size of a hypothesis that is consistent with the a sample. While this notion of approximation might seem unnatural, in the subsequent section, we turn our attention to approximating the solution of combinatorial optimization problems, for which it is natural to approximate some optimal scalar value while satisfying certain constraints.

4.1 Quantum advantage for combinatorial optimization

We now show a super-polynomial quantum advantage for approximating the solution of the combinatorial optimization task of formula coloring. We have already established the classical approximation hardness of FC-RSA in Theorem 3.5 and give a polynomial-time quantum algorithm for approximating FC-RSA in the proof of the following theorem.

Theorem 4.3 (Quantum efficiency for FC-RSA)**.** *There exists a polynomial-time quantum algorithm that, on input an instance F_S of* FC-RSA, *finds a valid coloring P such that*

$$|P| \leq opt_{\mathrm{FC}}(F_S)^{\alpha}$$

for all F_S and for some $\alpha \geq 1$.

Proof. Let us first describe how any instance F_S of FC-RSA looks like. The overview of the construction of FC-RSA is that we started from class C-RSA of log-depth poly-size Boolean circuits that explicitly decrypt an RSA ciphertext. The representation descriptions in C-RSA were then transformed using τ_1, τ_2 and τ_3 to the class DFA-RSA. Thus, recall that any instance S of $Con(\mathrm{DFA\text{-}RSA}, \mathrm{DFA})$ is of the form

$$S = \left\{ \left(\underbrace{\big\Vert_{l=1}^{p(n)} \mathrm{binary}\left(\mathrm{powers}_N(\mathrm{RSA}(x_i, N, e)), N, e\right)}_{=w_i,\text{ of length } p(n)\times(n^2+2n) \text{ bits}}, \underbrace{LSB(x_i)}_{=b_i} \right) \mid i = 1, \ldots, m \right\}, \tag{47}$$

where $||$ is the big concatenation of binary strings. Note that the repetition of w_i $p(n)$ times comes from the reduction τ_3, where for the construction of a DFA that simulates a log-space TM, the input needs to be repeated $p(n)$ times.

Now, F_S is obtained by the reduction (τ_4, g_1) from Section 3.2 and F_S is over the variables z_i^j, $1 \leq i \leq m$, $1 \leq j \leq p(n) \times (n^2 + 2n) + 1$. Recall that z_i^j encodes the state the DFA is in after reading bit j on input w_i. By the construction of F_S, we know that for each i_1, i_2 and j_1, j_2, such that

$$0 \leq j_1, j_2 < p(n) \times (n^2 + 2n) + 1 \tag{48}$$

and $w_{i_1}^{j_1+1} = w_{i_2}^{j_2+1}$, the following predicate

$$((z_{i_1}^{j_1} = z_{i_2}^{j_2}) \to (z_{i_1}^{j_1+1} = z_{i_2}^{j_2+1})) \tag{49}$$

occurs in F_S. Note that z_i^0 is the starting state of the DFA.

Consider the bit $w_1^{n^2+n}$ which is the least significant bit of N, for which we know that $LSB(N) = w_1^{n^2+n} = 1$, since N cannot be even. We know that for all other bits $w_{i_2}^{j_2+1}$ in S that are equal to $w_1^{n^2+n}$, there occurs a predicate of the form

$$((z_1^{n^2+n-1} = z_{i_2}^{j_2}) \to (z_1^{n^2+n} = z_{i_2}^{j_2+1})) \tag{50}$$

in F_S. Thus, by parsing F_S and looking for all predicates of the form as in (50), we can infer all bits in w_i given F_S, for all i. Thus, we can reconstruct all w_i's from F_S. Algorithm 2 does exactly this and runs in time poly(n), since there are $O(|S|^2)$ many clauses in F_S.

Algorithm 2: Infer w_i given F_S

```
Input  : An instance $F_S$ of FC-RSA, index $i \in \{1, \ldots, m\}$
Output : The bit string $w_i$
  // Initialize $w_i$ to the all 0 string
  $w_i \leftarrow 0^{(p(n) \times (n^2+2n))}$;
  // Set the first LSB(N)
  $w_i[n^2+n] \leftarrow 1$;
  // Set all other bits of $w_i$ that are also 1
  for $((z_1^{n^2+n-2} = z_i^j) \to (z_1^{n^2+n-1} = z_i^{j+1}))$ in $F_S$ do
    $w_i[j+1] \leftarrow 1$;
  end for
  return $w_i$;
```

Remember that our goal in this proof it to give a polynomial-time quantum algorithm that on input an F_S finds a valid coloring of size less than $opt_{\text{FC}}(F_S)^\alpha$. At this point we have described how the instances F_S look like and how we can extract the w_i's from it. After having obtained a w_i from F_S we read e and N from it and then construct the Boolean circuit c, by the same technique employed in Algorithm 1. It is important to note that c is exactly of the form of Boolean circuits in C-RSA, from which we originally constructed FC-RSA. When presented the input binary $(\text{powers}_N(\text{RSA}(x_i, N, e)), N, e)$, c outputs $LSB(x_i)$. We can transform c into a DFA that is consistent with S and then find a coloring for F_S from that DFA. Therefor, to obtain a DFA that is consistent with S, we run c through the instance transformations $t' = \tau_3(\tau_2(\tau_1(c)))$ to obtain the DFA t' which is consistent with S and of size poly(n). On input w_i, t' accepts if $LSB(x_i) = 1$ and rejects if $LSB(x_i) = 0$. Now we minimize t' using the standard DFA minimization algorithm [13] to obtain the smallest and unique DFA t which accepts the same language as t' and thus is also consistent with S and of minimal size. This DFA minimization is in principle not needed for the proof, but it is a further optimization step.

We then run Algorithm 3 to obtain a coloring for F_S from t. The DFA t consists of the set of states Q, the set of input symbols $\Sigma = \{0, 1\}$, the set of accepting states $\omega \subseteq Q$, the start state $q_0 \in Q$, and the transition function λ that takes as arguments a state and an input symbol and returns a state [13]. Furthermore, without loss of generality, we fix an ordering of the states in $Q = 0, \ldots, k-1$ with $q_0 = 0$. We can convince ourselves that the result of Algorithm 3 is indeed a valid coloring for F_S, since it assigns $z_{i_1}^{j_1}$ and $z_{i_2}^{j_2}$ the same color if and only if t is in the same state after reading $w_{i_1}^{j_1}$ on input w_{i_1} and after reading $w_{i_2}^{j_1}$ on input w_{i_2}. Therefore, a conjunct

$$((z_{i_1}^{j_1} = z_{i_2}^{j_2}) \to (z_{i_1}^{j_1+1} = z_{i_2}^{j_2+1})) \tag{51}$$

cannot be violated since it appears in F_S only if $w_{i_1}^{j_1+1} = w_{i_2}^{j_2+1}$ and by Algorithm 3, if $z_{i_1}^{j_1}$ is assigned the same color as $z_{i_2}^{j_2}$, then $z_{i_1}^{j_1+1}$ and $z_{i_2}^{j_2+1}$ have the same color [15]. Additionally, a

Algorithm 3: Obtain coloring for F_S from t

```
Input  : The inputs $w_i$ of $S$ and a DFA $t = (Q, \Sigma, \omega, q_0, \lambda)$ that is consistent with $S$
Output  A valid coloring for $F_S$
:
  // Initialize data structure
  $k \leftarrow |Q|$;
  $T \leftarrow$ Map{Int, Set};
  $T[0, \dots, k-1] \leftarrow \{\}$;
  // Read $w_i$ bit by bit, walk through $t$ and add fill the colors
  for $i = 1$ to $m$ do
    // Begin by the starting state
    $c \leftarrow 0$;
    $T[c] \leftarrow T[c] \cup \{z_i^0\}$;
    for $j = 1$ to $p(n) \times (n^2 + 2n)$ do
      $c \leftarrow \lambda(c, w_i^j)$;
      $T[c] \leftarrow T[c] \cup \{z_i^j\}$;
    end for
  end for
  return $\{T[0], \dots, T[k-1]\}$;
```

conjunct

$$(z_{i_1}^{p(n)\times(n^2+2n)} \neq z_{i_2}^{p(n)\times(n^2+2n)}) \tag{52}$$

cannot be violated since it appears only if $b_{i_1} \neq b_{i_2}$ and if $z_{i_1}^{p(n)\times(n^2+2n)}$ would be assigned the same color as $z_{i_2}^{p(n)\times(n^2+2n)}$, then t would be in the same state after reading all bits of w_{i_1} and w_{i_2}, which is either an accepting or rejecting state, which in turn contradicts that t is consistent with S and $b_{i_1} \neq b_{i_2}$ [15]. It follows that the coloring obtained through Algorithm 3 is upper bounded by $opt_{\mathrm{FC}}(F_S)^\alpha$ for some α, since t has polynomial size with the number of states given by $k = |Q| \leq n^\alpha \leq opt_{Con}(S)^\alpha = opt_{\mathrm{FC}}(F_S)^\alpha$ and Theorem 3.4. □

Thus, due to Theorem 3.5 and 4.3 we have the super-polynomial quantum advantage for approximating a combinatorial optimization solution. We reuse the techniques employed above to prove the super-polynomial quantum advantage for approximating the optimal solution of an integer linear programming problem, namely $\mathrm{ILP}_{F_S} \in$ ILP-RSA.

Theorem 4.4 (Quantum efficiency for ILP-RSA)**.** *There exists a polynomial-time quantum algorithm that, on input an instance* ILP_{F_S} *of* ILP-RSA*, finds a variable assignment A that satisfies all constraints and for which the objective function is bounded as*

$$\sum_{1\leq i\leq M} w_i \leq opt_{\mathrm{ILP}}(\mathrm{ILP}_{F_S})^\alpha$$

for all ILP_{F_S} *and for some $\alpha \geq 1$.*

Proof. Given an instance ILP_{F_S}, one can easily reconstruct F_S from the constraints (24) - (28) in polynomial time. It is then possible to obtain a valid coloring P of F_S given the routine described in the proof for Theorem 4.3, such that $|P| \leq opt_{\mathrm{FC}}(F_S)$. With P, we can get a valid assignment of the variables in ILP_{F_S} using the routine described in the $\Longrightarrow$-direction in the proof of Theorem 3.7. Also due to Theorem 3.7, we know that this variable assignment admits the objective function of ILP_{F_S} to be less than $opt_{\mathrm{ILP}}(\mathrm{ILP}_{F_S})^\alpha = opt_{\mathrm{FC}}(F_S)^\alpha$. □

Thus, due to the classical approximation hardness from Theorem 3.8, we encounter a super-polynomial quantum advantage for approximating the solution of an integer linear programming problem. It is important to stress that the reduction is explicit: That is to say, we can construct the instances for which one can achieve a quantum advantage of this kind.

5 The optimization problem in terms of a quantum Hamiltonian

The quantum algorithm presented is distinctly not of a variational type, as they are commonly proposed for approximating combinatorial optimization tasks using a quantum computer [9]. That said, it is still meaningful to formulate the problem at hand as an energy minimization problem, to closely connect the findings established here to the performance of variational quantum algorithms [6, 19] in near-term quantum computing, as this is the context in which such problems are typically stated. Such problems are commonly stated as *unconstrained binary optimization problems* of the form

$$x^* \quad := \quad \text{argmin}_{x\in\{0,1\}^n} f(x), \tag{53}$$

where $f : \{0,1\}^n \to \mathbb{R}$ is an appropriate cost function and x^* is a solution bit string of f. Particularly common are *quadratic unconstrained binary optimization problems*,

$$\begin{aligned} \text{minimize } f(x) \quad &= \quad x^T Q x, & (54)\\ x^* \quad &:= \quad \text{argmin}_{x\in\{0,1\}^n} f(x), & (55) \end{aligned}$$

where $Q = Q^T$ is a real symmetric matrix. In fact, it is a well-known result that all higher order polynomial binary optimization problems can be cast into the form of such a quadratic unconstrained binary optimization problem, possibly by adding further auxiliary variables; but it can also be helpful to keep the higher order polynomials. All such problems can be directly mapped to Hamiltonian problems. Notably, for quadratic unconstrained binary optimization problems, the minimum is equivalent with the ground state energy of the *quantum Ising Hamiltonian* defined on n qubits as

$$H = \sum_{i,j=1}^{n} Q_{i,j}(\mathbb{1} - Z_i)(\mathbb{1} - Z_j), \tag{56}$$

where Z_j is the Pauli-Z operator supported on site labeled j. For higher order polynomial problems, one can proceed accordingly.

Let us, pars pro toto, show how the formula coloring problem in the centre of this work can be cast into a quartic binary optimization problem. Let $k \in \mathbb{N}$ be an upper bound to the number of colors used for a formula over m variables, with $z_1, \dots, z_m \in \{1, \dots, k\}$ being the variables in the formula. We can then make use of $n = mk$ bits (which then turn into $n = mk$ qubits). These bits referred to as $b_{v,c}$ feature the double labels (v, c), where $v \in \{1, \dots, m\}$ labels the vertices and $c \in \{1, \dots, k\}$ the colors. If the vertex v is assigned the color c, we set $b_{v,c} = 1$, and $b_{v,d} = 0$ for all $d \neq c$. To make sure that the solution will satisfy such an encoding requirement, one adds a penalty of the form $(1 - \sum_{c=1}^{k} b_{v,c})^2$. The clauses of the form $(z_i \neq z_j)$ are actually precisely like in the graph coloring problem [28]. This can be incorporated by penalty terms of the type $\sum_{c=1}^{k} b_{z_i,c} b_{z_j,c}$: Then equal colors are penalized by energetic terms. The second type of clause requires more thought: Exactly if $(z_i = z_j)$ is true and $(z_j = z_k)$ is false, there should be a Hamiltonian penalty. As such, this is a quartic Boolean constraint of the form

$$\Big(\sum_{c=1}^{k} b_{z_i,c} b_{z_j,c}\Big)\Big(1 - \sum_{d=1}^{k} b_{z_j,d} b_{z_k,d}\Big). \tag{57}$$

Again, this can be straightforwardly be incorporated into a commuting classical Hamiltonian involving only terms of the type $(\mathbb{1} - Z_j)$ for suitable site labels j, precisely as commonly considered in quantum approximate optimization [9]. Lastly, to ensure we find a minimal coloring, we can either run the quantum optimization algorithm for increasing k and check whether a valid coloring has been found or one adds additional k qubits w_c, $c \in \{1, \dots, k\}$, which we enforce to be 1 if color c is used and 0 if color c is not used by adding the energetic penalty $b_{v,c} - b_{v,c} w_c$ for all $v \in \{1, \dots, m\}, c \in \{1, \dots, k\}$. This corresponds to enforcing the inequality $b_{v,c} \leq w_c$. We can then add the energetic penalty $\sum_{c=1}^{k} w_c$ to enforce the optimization algorithm to find the minimal coloring. For these reasons, the approximation results proven here motivate the application of quantum optimization techniques for commuting Hamiltonian optimization problems. Note that this construction is very similar to the integer linear program we proposed in Section 3.3 to reduce the formula coloring problem to ILP.

6 Discussion and outlook

In this work, we have addressed the important question of what potential quantum computers may offer for approximating the solution of combinatorial optimization problems. Given the importance of such problems in a wealth of applications and the large body of the recent literature on near-term quantum computing focusing on use cases of this kind, this is a reasonable thought.

In this work, we actually address this question from a fresh perspective. Equipped with tools from mathematical cryptography, we prove a super-polynomial speedup for approximating the solution of instances of NP-hard combinatorial optimization problems using a fault tolerant quantum computer. We explicitly show such speedups for instances of formula coloring and the much discussed integer linear programming which are proven to be hard to approximate. It would be an interesting line of thought to retrieve similar results for weak learning of representation classes, where the consistency constraint of h is relaxed to $1/\text{poly}(n)$-consistency.

In our work we answer the question of the potential of quantum computers to achieve substantial speed-ups for combinatorial optimization problems to the affirmative. In particular, we show that quantum computers can efficiently approximate combinatorial optimization problems that cannot be efficiently approximated by classical computations up to the same accuracy. This can be seen as an encouraging result for a family of quantum algorithms that could derive new applications from Shor's algorithm. It goes without saying that our findings cannot be used to make strong claims of a similar potential for variational quantum computations: Here, hard methods development is still required to come to such conclusions. At the same time, our work does provide guidance where to look for when reasonably assessing what is in store for quantum computers for this important class of problems. The work here shows what one can reasonably hope for when discussing the potential of variational quantum algorithms to tackle problems of combinatorial optimization.

7 Acknowledgements

This work has been supported by the Einstein Foundation (Einstein Research Unit on Quantum Devices), for which this is a joint node project, and the MATH+ Cluster of Excellence. It has also received funding from the BMBF (Hybrid), the BMWK (EniQmA), the Munich Quantum Valley (K-8), and the DFG (CRC 183). The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the programme of "Souverän. Digital. Vernetzt." Joint project 6G-RIC, project identification number: 16KISK030. Finally, it has also received funding from the QuantERA (HQCC).

A Modelling logical clauses as inequality constraints

In this appendix, we present some details of proofs that are made reference to in the main text.

A.1 Logical OR

To model the logical Boolean operator $\vee$, such that $s := (a \vee b)$ for binary variables s, a, b, we require the inequality constraints

$$s \geq a, \tag{58}$$
$$s \geq b, \tag{59}$$
$$s \leq a + b, \tag{60}$$

which is easily seen as being equivalent.

A.2 Logical equivalence

We are interested in modelling logical equivalences of the form $(a = 1) \iff (\hat{z}_u = \hat{z}_v)$ and $(b = 1) \iff (\hat{z}_u \neq \hat{z}_v)$ for the binary variables a, b and integers $\hat{z}_u, \hat{z}_v$. For the former, we model the forward and backward implications as follows.

$(a=1) \implies (\hat{z}_u = \hat{z}_v)$: Choose a large enough constant L such that $\hat{z}_u + \hat{z}_v \leq L$, then, since $\hat{z}_u, \hat{z}_v \geq 0$, the following constraints encode the implication.

$$\hat{z}_u \leq \hat{z}_v + (1-a)L, \tag{61}$$
$$\hat{z}_u \geq \hat{z}_v - (1-a)L. \tag{62}$$

Clearly, the constraints (61), (62) are satisfied for $\hat{z}_u = \hat{z}_v$ if $a = 1$ and for any $\hat{z}_u, \hat{z}_v$ if $a = 0$.

$(\hat{z}_u = \hat{z}_v) \implies (a=1)$: Note that this implication is equivalent to $(a \neq 1) \implies (\hat{z}_u \neq \hat{z}_v)$, which again is equivalent to $(a \neq 1) \implies ((\hat{z}_u > \hat{z}_v) \vee (\hat{z}_u < \hat{z}_v))$, which we will model below. We introduce a new binary variable q, for which, if $a = 0$ and $q = 1$ then $\hat{z}_u < \hat{z}_v$ and if $a = 0$ and $q = 0$ then $\hat{z}_u > \hat{z}_v$. This can be modelled by the constraints

$$\hat{z}_u < \hat{z}_v + (1-q+a)L, \tag{63}$$
$$\hat{z}_u > \hat{z}_v - (q+a)L. \tag{64}$$

The constraints (63), (64) are satisfied for $\hat{z}_u \neq \hat{z}_v$ if $a = 0$ and for any $\hat{z}_u, \hat{z}_v$ if $a = 1$. The variable q essentially indicates if $\hat{z}_u < \hat{z}_v$ or if $\hat{z}_u > \hat{z}_v$ when $a = 0$ and can be ignored after the optimization process. In a similar manner to the constraints above, we can model $(b=1) \iff (\hat{z}_u \neq \hat{z}_v)$ as

$$\hat{z}_u < \hat{z}_v + (2-q'-b)L, \tag{65}$$
$$\hat{z}_u > \hat{z}_v - (1+q'-b)L, \tag{66}$$
$$\hat{z}_u \leq \hat{z}_v + bL, \tag{67}$$
$$\hat{z}_u \geq \hat{z}_v - bL, \tag{68}$$

in terms of inequality constraints.

References

[1] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM J. Comp.*, 17:194–209, 1988. DOI: 10.1137/0217013.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal, W. Cook, D. G. Espinoza, M. Goycoolea, and K. Helsgaun. Certification of an optimal TSP tour through 85.900 cities. *Oper. Res. Lett.*, 37:11–15, 2009. DOI: 10.1016/j.orl.2008.09.006.

[3] F. Arute et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019. DOI: 10.1038/s41586-019-1666-5.

[4] P. W. Beame, S. A. Cook, and H J. Hoover. Log depth circuits for division and related problems. *SIAM J. Comp.*, 15:994–1003, 1986. DOI: 10.1137/0215070.

[5] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's Razor. *Inf. Proc. Lett.*, 24:377–380, 1987. DOI: 10.1016/0020-0190(87)90114-1.

[6] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles. Variational quantum algorithms. *Nature Rev. Phys.*, 3:625–644, 2021. DOI: 10.1038/s42254-021-00348-9.

[7] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial optimization*. Wiley, New York, 1997. DOI: 10.14760/OWR-2008-51.

[8] Y. Deller, S. Schmitt, M. Lewenstein, S. Lenk, M. Federer, F. Jendrzejewski, P. Hauke, and V. Kasper. Quantum approximate optimization algorithm for qudit systems with long-range interactions. 2022. arXiv:2204.00340.

[9] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. 2014. arXiv:1411.4028.

[10] O. Goldreich. *Foundations of cryptography, Volume 2*. Cambridge University Press, Cambridge, 2004. DOI: 10.1017/CBO9780511546891.

[11] G. González-García, R. Trivedi, and J. I. Cirac. Error propagation in NISQ devices for solving classical optimization problems. 2022. arXiv:2203.15632.

[12] D. Hangleiter and J. Eisert. Computational advantage of quantum random sampling. 2022. arXiv:2206.04079.

[13] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Pearson Deutschland, 2013. DOI: 10.1145/568438.568455.

[14] J. Hromkovič. *Algorithmics for hard problems: Introduction to combinatorial optimization, randomization, approximation, and heuristics*. Springer, Berlin, 2004. DOI: 10.1007/978-3-662-04616-6.

[15] M. J. Kearns and L. G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Machine learning: From theory to applications*, pages 29–49, 1993. DOI: 10.1145/174644.174647.

[16] M. J. Kearns and U. Vazirani. *An introduction to computational learning theory*. MIT press, Cambridge, MA, 1994. DOI: 10.7551/mitpress/3897.001.0001.

[17] J. Liu, F. Wilde, A. A. Mele, L. Jiang, and J. Eisert. Noise can be helpful for variational quantum algorithms. 2022. arXiv:2210.06723.

[18] Y. Liu, S. Arunachalam, and K. Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Phys.*, 17:1013, 2021. DOI: 10.1038/s41567-021-01287-z.

[19] J. R McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New J. Phys.*, 18:023023, 2016. DOI: 10.1088/1367-2630/18/2/023023.

[20] A. Montanaro. Quantum algorithms: an overview. *npj Quant. Inf.*, 2:15023, 2016. DOI: 10.1038/npjqi.2015.23.

[21] N. Pirnay, R. Sweke, J. Eisert, and J.-P. Seifert. A super-polynomial quantum-classical separation for density modelling. 2022. arXiv:2210.06723.

[22] J. Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018. DOI: 10.22331/q-2018-08-06-79.

[23] Y. Quek, D. Stilck França, S. Khatri, J. J. Meyer, and J. Eisert. Exponentially tighter bounds on limitations of quantum error mitigation. 2022. arXiv:2210.11505.

[24] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21:120–126, 1978. DOI: 10.1145/359340.359342.

[25] P. W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proc. 35th Ann. Symp. Found. Comp. Sc.*, pages 124–134. Ieee, 1994. DOI: 10.1109/SFCS.1994.365700.

[26] D. Stilck Franca and R. García-Patrón. Limitations of optimization algorithms on noisy quantum devices. *Nature Phys.*, 17:1221, 2020. DOI: 10.1038/s41567-021-01356-3.

[27] R. Sweke, J.-P. Seifert, D. Hangleiter, and J. Eisert. On the quantum versus classical learnability of discrete distributions. *Quantum*, 5:417, 2021. DOI: 10.22331/q-2021-03-23-417.

[28] Z. Tabi, K. H. El-Safty, Z. Kallus, P. Hága, T. Kozsik, A. Glos, and Z. Zimborás. Quantum optimization for the graph coloring problem with space-efficient embedding. In *2020 IEEE Int. Conf. Quant. Compu. Eng. (QCE)*, pages 56–62, 2020. DOI: 10.1109/QCE49297.2020.00018.

[29] R. Takagi, S. Endo, S. Minagawa, and M. Gu. Fundamental limits of quantum error mitigation. *npj Quant. Inf.*, 8:114, 2022. DOI: 10.1038/s41534-022-00618-z.

[30] L. G. Valiant. A theory of the learnable. *Comm. ACM*, 27:1134–1142, 1984. DOI: 10.1145/1968.1972.

[31] D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge University Press, Cambridge, 2011. DOI: 10.1017/CBO9780511921735.

[32] L. A. Wolsey and G. L. Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999. DOI: 10.1002/9781118627372.

[33] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X*, 10:021067, 2020. DOI: 10.1103/PhysRevX.10.021067.