Integer Factorization through Func-QAOA

Mostafa Atallah[1], Haemanth Velmurugan[2], Rohan Sharma[3], Siddhant Midha[4], Shamim Al Mamun[5], Ludmila Botelho[6,7], Adam Glos[7], and Özlem Salehi[*7,8]

[1]Department of Physics, Faculty of Science, Cairo University, Giza 12613, Egypt
[2]Department of Computer Science and Engineering, NIT Tiruchirappalli, India
[3]Department of Applied Geophysics, IIT Dhanbad, India
[4]Department of Electrical Engineering, IIT Bombay, India
[5]Department of Electrical and Electronics Engineering, Islamic University of Technology, Gazipur, Dhaka 1704
[6]Joint Doctoral School, Silesian University of Technology, Akademicka 2A, Gliwice, Poland
[7]Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Poland
[8]QWorld Association, Tallinn, Estonia

Abstract

Integer factorization is a significant problem, with implications for the security of widely-used cryptographic schemes. No efficient classical algorithm for polynomial-time integer factorization has been found despite extensive research. Although Peter Shor's breakthrough quantum algorithm offers a viable solution, current limitations of noisy intermediate-scale quantum (NISQ) computers hinder its practical implementation. To address this, researchers have explored alternative methods for factorization suitable for NISQ devices. One such method is the Quantum Approximate Optimization Algorithm, which treats factoring as an optimization problem defined over binary bits, resulting in various problematic aspects. In this paper, we explore the Func-QAOA approach for factorization, which premises overcoming some of the limitations of previous approaches and allows the incorporation of more advanced factorization techniques. After reviewing the most promising quantum implementations for integer arithmetics, we present a few illustrative examples to demonstrate the efficacy of the Func-QAOA approach and discuss methods to reduce the search space to speed up the optimization process.

[*]ozlemsalehi@gmail.com

1 Introduction

Integer factorization is one of the most significant problems lying at the heart of modern cryptography. Widely used RSA cryptographic scheme [1] relies on the fact that no efficient classical algorithm is known that can be used to factor large integers in a feasible time. Despite extensive research, no polynomial-time algorithm has been discovered to solve the problem, yet it has not been definitively proven that no such algorithm exists. Classical algorithms like trial division, Fermat's factorization [2], and Pollard's rho algorithm [3] provide basic solutions to the problem but run in exponential time. More advanced algorithms like those employing elliptic curves [4] and general number field sieve [5] offer the best complexity known so far, requiring subexponential time.

The idea of quantum computers emerged in the 1980s, revolutionizing the field of computing by harnessing quantum phenomena to perform efficient calculations. Peter Shor made a groundbreaking discovery in 1994, putting forward a polynomial time algorithm to solve the factoring problem using quantum computers [6] and recently he was awarded the Breakthrough Prize in 2022 for this achievement [7]. The theoretical discovery was followed by experimental demonstrations of factoring number 15 using photonic and NMR qubits [8–10] and improvements of the resources required for implementation [11–13]. Despite the mentioned efforts, the largest integer that one could factor using Shor's algorithm is 21 [14] as decoherence has a destructive impact on noisy intermediate scale quantum (NISQ) [15] computers which are small in size. The current estimates on the number of required physical qubits to break RSA-2048 range from millions to billions [16], which is far from what we have today.

This has led researchers to seek alternative methods for solving factorization problem suitable for NISQ. Factorization problem has been targeted in the framework of adiabatic quantum computing (AQC) [17] in [18–22] and its gate-based counterpart quantum approximate optimization algorithm (QAOA) [23] in [24–26]. Those approaches require problem to be expressed as an optimization problem in the form of an Ising model. The natural cost function to minimize results from $f(x, y) = (m - xy)^2$, where $f(p, q) = 0$ is the minimum of the function given that $m = pq$ is the integer to be factorized. Alternatively, one can express integers using their binary representation and end up with a set of equations that need to be satisfied, which was first observed in the context of classical optimization in [27]. Then, the problem Hamiltonian can be obtained by replacing binary variables with the corresponding spin variables, and further with Pauli Z operators.

For AQC, one needs to come up preferably with a 2-local problem Hamiltonian, as higher-order models are experimentally harder to implement. Some classical simplification rules and a penalty method are developed by [19] to reduce the number of qubit requirements and to obtain a 2-local model. In [20], some additional classical preprocessing rules are used to simplify the constraints, resulting in a reduction in the number of required qubits, and the integer 143 is factored using only four qubits. In [21], the authors realize that the same 4-qubit Hamiltonian can be used for factoring the integers 3599, 11663, and 56153. Other relevant research includes those using quantum annealing [28, 29], which is a heuristic optimization algorithm running in

the framework of AQC [30, 31] and can be implemented on D-Wave quantum annealers [32]. A 40-bit integer was recently factored using a hybrid solver of D-Wave [33].

QAOA is used in the scope of factorization for the first time by Anschuetz et al. [24] under the name variational quantum factoring (VQF). A new set of classical preprocessing rules are presented, through which authors claim to reduce the number of qubit requirements to $\mathcal{O}(n_m)$ qubits where $n_m \sim \log m$. Building on the work of [24], further preprocessing rules are given in [25] together with the claim that a 40-bit integer is factored using a superconducting processor consisting of only three qubits. In a recent work by Yan et al. [26], QAOA is proposed to speedup Schnorr's algorithm which is based on classical lattice reduction, to obtain an algorithm that uses only $\mathcal{O}(n_m/\log n_m)$ qubits. The authors report that they have successfully factored a 48-bit integer using a 10-qubit quantum processor.

Along with these initial advancements in the field, the NISQ-era efforts for factorization have faced criticism from other researchers for various reasons. One line of criticism [34, 35] centers around the fact that most of the works use biprimes of special form [20, 21], which can be factored using only a few qubits after classical preprocessing and thus cannot be used for general biprimes. Thus, the resulting binary models are so simple that they are easily tractable classically. Moreover, the idea of VQF and the claims made in [25] and [26] encountered more significant criticisms, drawing reactions from various researchers, including Scott Aaronson [36, 37]. The primary criticism is directed towards VQF's treatment of the factoring problem as an elementary optimization task reduced to a satisfiability problem, overlooking any underlying mathematical structure.

In this paper, we concretize and highlight additional concerns and problematic aspects associated with approaching factorization as a satisfiability problem and propose the Func-QAOA approach for factorization to mitigate these drawbacks. First of all, as a consequence of expressing the problem as binary multiplication of integers, additional variables are generated like the carry-bits [24], even at the initial step of constructing a binary optimization model. This results in generation of redundant bits or even ones lacking interpretation with the original problem. The commonly used X-mixer in VQF further aggravates the problem, as it allows amplitude exchange between states which are conceptually unrelated in scope of the factoring problem. Finally, constructing the binary model from the simplistic approach of the product of two integers is unlikely to provide any speedup against classical algorithms. Taking into consideration the mentioned aspects, we describe a framework to utilize Func-QAOA for the factorization problem and analyze its potential to address and alleviate some of the aforementioned criticisms. Proposed in [38], Func-QAOA offers an alternative implementation for QAOA that eliminates the need for an Ising model representation of the problem. Although we do not give a concrete example that provides practical advantage, building upon this novel approach, we demonstrate how the proposed framework can remedy the aforementioned issues through the ingenious design of initialization, ansatz and mixer circuits. In addition, we investigate how the context of factorization can be incorporated into the optimization process through the use of more advanced factorization techniques.

The paper is organized as follows. In Sec. 2 we recall the factorization problem, QAOA and Func-QAOA. In Sec. 3 we present a few pedagogical examples on how one can employ Func-QAOA framework for factorization. In the same section we concretize and highlight concerns and problematic aspects and show how the proposed framework mitigates some of them. Later we present some methods to reduce the search space. In Sec. 4 we discuss the results presented in the paper and conclude.

2 Preliminaries

2.1 Factorization problem

Factorization problem if considered in the context of cryptographic applications, naturally becomes a search problem: given a composite integer m, the goal is to find a factors $\bar{p}, \bar{q}$ s.t. their product results in m. Among the set of solutions $\{(p,q) : 1 < p, q < n\}$ which we call the search space, we look for the correct solution $(\bar{p}, \bar{q})$. Given a particular solution (p, q), using the definition of the integer factorization problem, it is enough to certify whether $m = pq$. One can consider different search spaces and certificates for the very same problem. Some of those include:

1. Looking for (a, b) s.t. $a^2 - b^2 = m$, where the factors can be recovered as $\bar{p} = a - b$ and $\bar{q} = a + b$.

2. Looking for the smallest even r such that $a^r = 1 \mod m$ and $a^{r/2} \neq -1 \mod m$, where $a \leq m$ is an integer such that $gcd(a, m) = 1$, where the factors can be computed by $\gcd(a^{r/2} \pm 1, m)$.

3. More advanced search spaces and certifications like those based on the elliptic curves.

Note that the correct solution in general does not have to consist of the factors $\bar{p}$ and $\bar{q}$, but allows efficient construction of them.

Many of the algorithms targeting factorization problem work by taking a sample from the search space, which is then analysed to see whether it is a correct solution that allows to produce the factors. If this is the case, the algorithm ends, but otherwise, another sample is taken and the procedure is repeated. For this type of algorithms, we distinguish two critical properties. First one is the search space of the algorithm. Directly connected to this is the probability of obtaining the correct solution. For example, if we are sampling a candidate factor from $\{2, \ldots, \lfloor \sqrt{m} \rfloor\}$, the probability of finding the correct solution is $\approx \frac{1}{\sqrt{m}}$. Another important property of the algorithm is the certifying time t that is required for certifying the given input. Note that the total time for the algorithm to work will be the certifying time over the probability of finding the solution.

Currently no classical algorithm is known to have a total time that is polynomial; however, the underlying reason might may vary for each algorithm. For example, for the aforementioned

naive algorithm in which a potential factor p is sampled, the certifying time is polynomial – it is enough to verify if $m = 0 \mod p$. The problem with this algorithm lies in sampling the correct solution as the corresponding probability is low. On the other hand, one can consider more advanced search algorithms instead of random sampling. This opens a room for quantum search techniques like Grover's search [39, 40] that can accelerate classical algorithms and optimization techniques, which may provide the answer much faster than the classical counterparts.

An interesting class of algorithms are those which are based on elliptic curves. These algorithms have comparatively large, inverse of subexponential probability of finding the correct solution while the certifying time is also subexponential. Hence, the total time is subexponential as well, resulting in elliptic curve factorization methods which are among the best classical algorithms known so far. This was used to provide a candidate for the Grover-based fault-tolerant quantum algorithm [40].

2.2 Quantum Approximate Optimization Algorithm

Quantum Approximate Optimization Algorithm (QAOA) [23] is a NISQ-era variational algorithm for which the ansatz originates from the quantum adiabatic theorem [41]. For the tunable parameters $\theta_i^{\mathrm{mix}}, \theta_i^{\mathrm{obj}}$ for $i = 1, \ldots, m$, the optimized quantum state $|\theta^{\mathrm{mix}}, \theta^{\mathrm{obj}}\rangle$ takes the form

$$|\theta^{\mathrm{mix}}, \theta^{\mathrm{obj}}\rangle = \prod_{i=1}^{m} \exp\left(-\mathrm{i}\theta_i^{\mathrm{mix}} H_{\mathrm{mix}}\right) \exp\left(-\mathrm{i}\theta_i^{\mathrm{obj}} H\right) |\psi_0\rangle, \tag{1}$$

where H is a problem Hamiltonian, a diagonal Hamiltonian whose ground state encodes the optimal solution, and H_{mix} is the so-called mixer Hamiltonian which is responsible for transferring the amplitude between the computational basis states. $|\psi_0\rangle$ is a fixed initial quantum state, usually (but not necessarily) a ground state of H_{mix}. In the original QAOA, it was proposed to choose $|\psi\rangle$ as the uniform superposition and H_{mix} as the sum of 1-local Pauli-X operators, i.e. $H_{\mathrm{mix}} = -\sum_i X_i$. However, in subsequent research, various mixers have been proposed together with suitable initial states, which allow starting in a quantum state with a better overlap with the low energy states [42–45]. Owing to that development, the optimization may be performed in a much smaller space, possibly at a higher cost of the circuit depth or the number of gates. A particularly interesting is the Grover mixer [44], which requires a circuit that transforms $|0\rangle$ state into a $|\psi_0\rangle$ being a superposition of some set of solutions, and then reuses the same circuit to allow all-to-all amplitude transfer between the solutions present in $|\psi_0\rangle$. Note that the original mixer $-\sum_i X_i$ can be in turn interpreted as transferring amplitude along the hypercube of the set of all solutions, as X_i can be considered as a NOT operation acting on the i-th qubit.

Just as there is considerable freedom in choosing the mixer Hamiltonian and the initial state, there is flexibility in constructing the problem Hamiltonian. A typical choice is to start with a quadratic unconstrained binary optimization (QUBO) formulation which then can be transformed into 2-local Ising model. A plethora of QUBO formulations for various important paradigmatic problems were proposed in [46]. Subsequently, higher-order binary optimization

(HOBO) formulations were proposed as a remedy for the possibly large number of qubits [47,48]. Contrary to QUBO, HOBO is an arbitrary order pseudo-Boolean polynomial. In this case, it has to be guaranteed that the corresponding Ising model has a polynomial number of terms, to ensure its feasibility for execution on a quantum computer. Alternatively, for the Max-K-Cut and Knapsack problems, QAOA ansatz constructions have been proposed without giving the explicit QUBO or HOBO formulation [49,50]. The idea is generalized and formally introduced in [38] under the name Func-QAOA.

FUNC-QAOA begins with an abstract description of the problem, eliminating the need for an Ising model representation. In many of the combinatorial optimization tasks, the problem involves minimization of an objective function subject to some constraints, which makes up the problem definition. For the objective, a classical program is designed to compute the objective value, and for each constraint, a classical program is designed that outputs a positive number if the constraint is not satisfied and 0 otherwise. These programs are then translated into quantum circuits and combined with a rotation around the Z axis to achieve an equivalent effect to term-by-term implementation of the conventional problem Hamiltonian implementation of QAOA. The FUNC-QAOA framework enables a direct mapping from abstract problem descriptions to quantum circuits, greatly increasing the number of problem Hamiltonians that can be implemented efficiently. Furthermore, it is demonstrated that near-optimal circuits in terms of e.g. number of qubits and gates can be obtained for Travelling Salesman Problem and Max-K-Cut using FUNC-QAOA [38].

As an example, let us consider the graph coloring problem which is a decision problem asserting whether K colors are sufficient to color the nodes of a particular graph $G = (V, E)$, so that adjacent nodes are colored with different colors. Since this is a decision problem, it is natural to define it as a set of constraint s.t. for each edge $\{u, v\} \in E$ we have $c(u) \neq c(v)$ where $c : V \to \{0, \ldots, K-1\}$ is the coloring map. Since coloring maps form a natural search space for this problem, we require $|V|$ registers each with $\lceil \log K \rceil$ qubits so that v-th register with state $|c(v)\rangle$ encodes the color $c(v)$ (in binary encoding) of the node $v \in V$. Then a particular way of asserting that c is a proper coloring is to follow the steps listed below for each edge $\{u, v\} \in E$:

1. On register $|c(v)\rangle$ store the XOR (bit-wise addition modulo 2) of $|c(v)\rangle$ and $|c(u)\rangle$ which can be done with $\lceil \log K \rceil$ CNOTs.

2. Apply X gate (NOT) on all the qubits of $|c(v)\rangle$; if the edge connects nodes with the same color, register $|c(v)\rangle$ will be just $|1 \ldots 1\rangle$.

3. Implement multi-controlled NOT operation with control on $|c(v)\rangle$ and target on additional auxilliary qubit $|\text{flag}\rangle$ which is initially set to $|0\rangle$.

4. Apply Z-rotation on the $|\text{flag}\rangle$ qubit with an magnitude proportional to the trained parameter θ^{obj}, which introduces the phase for the input $|1\rangle$.

5. uncompute steps 1.-3.

6

The problem Hamiltonian corresponding to the above procedure takes the form

$$\sum_{\{u,v\}\in E} [c(u) = c(v)],\tag{2}$$

where $[\varphi]$ is the Iverson notation, which can be used to estimate the energy of the quantum state. The function of the form above that allows to compute the energy of the measured solution is called a constraint function. Note that for the particular map c, the constraint function gives value 0 if and only if all pairs of adjacent nodes are coloured differently.

3 FUNC-QAOA for integer factorization

In this section, we show how the FUNC-QAOA framework can lead to an interesting candidate for integer factorization. In Sec. 3.1, we demonstrate the idea through pedagogical examples. Then in Sec. 3.2, we analyze the problematic aspects of solving integer factorization with QAOA and discuss potential benefits achievable by using the proposed framework. Finally, in Sec. 3.3, we include technical remarks on reducing the search space.

3.1 Demonstration of Func-QAOA through pedagogical examples

Designing a FUNC-QAOA circuit involves 4 main steps: Providing a problem description involving constraints and the objective function (if exists), writing classical programs for the constraints and the objective function, implementing the corresponding circuits and choosing appropriate initial state and mixer implementation. In this subsection we show examples for all the steps based on the multiplication, modulo, and order-finding certificates.

We will first consider the search space defined as all possible pairs of factors of m, namely $\{(p,q) : 1 < p \leq \sqrt{n} \leq q \leq n/2\}$. The goal is to find the pair $(\bar{p}, \bar{q})$ such that $m = \bar{p}\bar{q}$. For this particular problem definition, we have a single constraint $m = pq$. In the given search space, there is only one pair of integers which satisfy the constraint. The Func-QAOA algorithm works as follows. The two registers $\mathcal{H}_p$ and $\mathcal{H}_q$ are initialized to a particular initial state (this aspect will be considered later in this section). Then the constraint is implemented by taking the following steps:

1. Using an arbitrary implementation of the multiplication (possibly with some auxiliary qubits), store the product of p and q from $\mathcal{H}_p$ and $\mathcal{H}_q$ (in computational basis) in an additional subsystem $\mathcal{H}_{\text{product}}$ which is initially set to $|0\ldots0\rangle$.

2. Modify the state of $\mathcal{H}_{\text{product}}$ by taking bit-wise XOR with m.

3. Apply rotation in Z basis on the $\mathcal{H}_{\text{product}}$ proportional to the trained parameter.
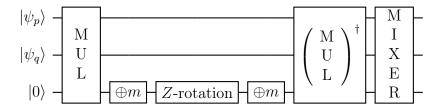
4. Uncompute steps 1.-2.

Figure 1: Single layer of the Func-QAOA for multiplication-based certification.

The corresponding quantum circuit can be found in Fig. 1. The number of qubits in $\mathcal{H}_{\text{product}}$ should be sufficiently large so that any product of p and q that can be produced can be stored. Note that the state of $\mathcal{H}_{\text{product}}$ after step 2. is equal to $|0\dots0\rangle$ iff the computational basis states of $\mathcal{H}_p$ and $\mathcal{H}_q$ are factors of m.

The framework above has a few degrees of freedom. First of all, there is a plethora of multiplication algorithms one can choose. Those differ in particular in the number of auxiliary qubits, depth, number of gates, and other aspects that might be useful when considering Func-QAOA in NISQ-era computing, see Table. 1 for reference. The choice of the particular algorithm is irrelevant to the Func-QAOA itself and can be done based on the properties of the quantum hardware at hand. However, we can see that depending on the quantum hardware at hand, one can choose implementations optimizing the number of auxiliary qubits like the ones presented in [51–53], or number of gates [52, 54]. Especially for the number of gates, we can observe a provable decrease in the number of CNOTs: it becomes strictly smaller than $\mathcal{O}(n^2)$, which is typically observed for dense QUBO formulations.

Another degree of freedom is the choice on how the phase to wrong solutions is added. For the above, a natural choice is to apply qubit-wise rotations, which would result in the phase proportional to the number of bits at which pq and m differ. For this Hamiltonian, the ground state matches the correct solution $(\bar{p}, \bar{q})$. Alternatively, one can perform further operations to compute $(m - pq)^2$ or $|m - pq|$ and apply the phase proportional to that value. However in here because the numbers are exponentially large, an exponentially precise rotation would need to be used. Furthermore, this may require exponentially many shots from the quantum hardware to bring sufficient estimation of the energy.

Finally, there is a freedom in the choice of the initial state and the corresponding mixer. We believe that particularly interesting choices rely on the Grover mixer [44]. Grover mixer uses the initial state preparation circuit, to allow all-to-all transition only between the solutions that has nonzero amplitude in the initial state. Since the problem Hamiltonian (whether implemented conventionally by trotterization, or computed as in Func-QAOA) modifies only the relative phase for each solution, Grover mixer guarantees to preserve the space of the solutions that have nonzero amplitude in the initial state. As for the initial state, one can consider starting in the uniform superposition of the search space; however, creating such state might be costly. Instead, one can just start by applying Hadamard gates on all the qubits in $\mathcal{H}_p$ and $\mathcal{H}_q$, which would results in the search space $\{(p, q) : 0 \leq p < 2^{\lceil \log \sqrt{m} \rceil}, 0 \leq q < 2^{\lceil \log(m/2) \rceil}\}$. This will at

	Qubits	Gates	Depth	Additional information
Rines et al. [51]	$\log n$	$*n^2$	n	Discusses Montgomery and Barrett reduction techniques; QFT-based.
	n	n^2	$n \log n$	Uses prefix carry lookahead adders.
	n	n^2	n^2	Uses ripple adders
Gidney [52]	n	$n^{\log 3}$	$n^{\log 3 - 1} \log^2 n$	Uses divide and conquer approach; has high overheads and is more efficient than schoolbook multiplication at around 10000 bits.
Dutta et al. [54]	$n^{1.404}$	$n^{\log_6 16}$	$n^{1.143}$	Toom-2.5 algorithm; performs better than naive multiplication beyond 300 bits.
Larasati et al. [55]	$n^{1.353}$	n^2	$n^{1.112}$	Toom-3 algorithm.
Babu et al. [56]	n^2	n^2	n	Optimal tree-based multiplication technique.
Álvarez-Sánchez et al. [53]	n	—	$\log^2 n$	Quantum version of Booth's algorithm.

Table 1: Review of quantum implementations for integer multiplication. Only the leading terms, without the constants are presented. '$*$' indicates that exponential-precision gates are required, which is likely not practical for NISQ-era quantum computing. Note that we refer to number of auxiliary qubits by qubits.

most quadruple the size of the search space but allow much more efficient implementation of the initial state and mixer.

Note that with Func-QAOA, we were able to naturally reduce the search space from $\mathcal{O}\left(\log^2 m\right)$ to $\mathcal{O}(\log m)$, by avoiding the introduction of carry bit variables as done in [24]. However, with Func-QAOA, one can further reduce the search space by exploring the alternative search space choices. Note that if one obtains the smaller prime factor $\bar{p}$, then obtaining the second one $\bar{q} = m/\bar{p}$ is straightforward. This observation suggests that approaching the factorization problem by searching for pairs of products is redundant, and a modulo operation could be used instead, i.e. a certification of the form m MOD $p \equiv 0$. The search space size is defined as the logarithm of the number of computational basis vectors spanning the quantum state during the optimization. Note that if we seek for the smaller factor, the search space size is reduced from $\sim \log_2(\sqrt{m}) + \log_2(m/2)$ to $\sim \log(\sqrt{m}) = \frac{1}{2}\log m$, resulting in $\approx 66\%$ improvement in the search space size. On the other hand, this may require a more advanced circuit for implementing modulo operation, which might be more costly in the number of quantum resources. The cost of such operation is presented in Table 2. While we were able to signifi-

	Qubits	Gates	Depth	Additional information
Şahin [57]	3	$*n^2 + m^2$	n^2	QFT-based approach for signed integers with n-bit dividend and m-bit divisor.
Thapliyal et al. [58]	0	n^2	n^2	A general framework for division using adder and subtractor blocks.
Jamal et al. [59]	n^2	n^2	n^2	Reversible divider hardware implemented on conventional and high-speed division arrays.

Table 2: Review of quantum implementations for modulo operation. Only the leading terms, without the constants are presented. '$*$' indicates that exponential-precision gates are required, which is likely not practical for NISQ-era quantum computing. Note that we refer to number of auxiliary qubits by qubits.

cantly reduce the search space size, the number of gates required for modulo operation brings the number of CNOTs required back to $\mathcal{O}(n^2)$, which is the worst-case scenario in complexity for n-bit QUBO formulation.

Since FUNC-QAOA relies on universality of the gate-based model, one can design more complicated search spaces than the ones considered above. For instance, in principle, FUNC-QAOA can be used for order finding, which is the main ingredient of Shor's algorithm. We will demonstrate this only for pedagogical purposes, to further clarify how FUNC-QAOA works. Let $x < m$ be a number that is coprime with m. The smallest integer r satisfying the relation $x^r = 1 \mod m$ is called the order. Knowing the order, one can find out the factors through simple calculations, provided that the order is even and that it satisfies the relationship $x^{r/2} \neq -1 \mod m$. Note that this procedure is probabilistic as such r may not exist depending on the choice of x and in such a case, the procedure should be repeated with a different x. Since probability of choosing a valid x is constant it can be done classically and we push the difficulty of finding the correct solution $\bar{r}$ to the quantum part of the computation.

In this case, the search space consists of integers $\{r : 2 \leq r < m, r \text{ even}\}$ and we implement the constraint $x^r = 1 \mod m$. Note that we have an optimization problem, as we aim to find the smallest r. To design a FUNC-QAOA, we initialize register $\mathcal{H}_r$ to store $|r\rangle$ in equal superposition of all basis states and we choose a random x. The last qubit should be fixed to $|0\rangle$ as we are interested in even r only. We reserve another register $\mathcal{H}_o$ to store $|x^r \mod m\rangle$ where each qubit is initialized to $|0\rangle$. Next, we apply modular exponentiation circuit to store $|x^r \mod m\rangle$. Note that various circuit constructions appear in the literature for performing modular exponentiation [11–13] with trade-off between the number of qubits and the number of gates. We check if the result is equal to 1 and set an additional $|flag\rangle$ qubit to $|1\rangle$ if this is not the case. We apply rotation on $|flag\rangle$ proportional to the trained parameter. In addition,

	Qubits	Gates	Depth	Additional information
Draper [60]	0	$*n^2$	n	Performs modular addition; QFT-based.
	0	$n \log n$	$\log n$	Approximate QFT-based; [61] extends this to full (non-modular) addition.
Cuccaro et al. [62]	1	n	n	Ripple adder; [63] reduces the number of toffoli gates used with more ancilla qubits.
Thapliyal et al. [64]	0	n	n	Reversible ripple carry adder.
Draper et al. [65]	n	n	$\log n$	Carry lookahead adder.
Thapliyal et al. [66]	n	n	$\log n$	Reversible carry lookahead adder; claims slight improvement over [65] in terms of number of gates and depth.
Takahashi et al. [67]	$n/d(n)$	n	$d(n)$	$d(n) = \Omega(\log n)$; Combination of ripple and carry lookahead adder approaches [68]; the circuit can be transformed for linear nearest neighbor architecture without increasing the size or depth asymptotically; also discusses improvements when unbounded fanout gates are available.
Gidney [69]	n/b	$n + n/b - b$	$b + \log n/b$	Block lookahead adders; parallelizes across blocks of size b, instead of all bits.
Choi et al. [70]	$2n - \sqrt{n}$	—	$\sqrt{n}$	Adder for 2D NTC architecture.

Table 3: Review of quantum implementations for integer addition. Only the leading terms, without the constants are presented. '$*$' indicates that exponential-precision gates are required, which is likely not practical for NISQ-era quantum computing. Note that we refer to number of auxiliary qubits by qubits.

we apply rotation on $|r\rangle$ proportional to the magnitude of r, as we would like to minimize r. Finally, all the steps are uncomputed. Unfortunately, $r = 0$ is a trivial answer but additional energy can be incorporated for this specific case.

Note that the above proposals are agnostic to the actual implementation of the arithmetic operations. This is because Func-QAOA is indifferent to how the operations are executed, yet it would be desirable to have a noise-robust implementations to the greatest extent possible. Therefore, the specific implementation can be chosen based on the properties of the quantum

	Qubits	Gates	Depth	Additional information
Şahin [57]	0	$*n^2 + nm - m^2$	n^2	QFT-based approach for subtracting m-bit signed integer from an n-bit signed integer.
Thapliyal [71]	$n - 1$	$*n$	n	Extension of half and full subtractor in [72]; reversible subtractor.
	0	n	n	Subtractor based on adder; also discusses the design of adder-subtractors.

Table 4: Review of quantum implementations for integer subtraction. Only the leading terms, without the constants are presented. '$*$' indicates that exponential-precision gates are required, which is likely not practical for NISQ-era quantum computing. Note that we refer to number of auxiliary qubits by qubits.

hardware at hand, taking into account aspects like limited connectivity, number of qubits or quantum volume. To complement this subsection, we provide a review of existing implementations of various arithmetic operations, including addition as presented in Table 3, subtraction as presented Table 4, in multiplication as presented in Table 1, and modulo as presented in Table 2.

3.2 Addressing the concerns on QAOA for factorization

One of the main concerns regarding QAOA when used for solving the factorization problem is that the optimization algorithms are not suitable for the search or decision problems. Indeed, such algorithms need to find the *global optimum* of the objective function to provide the correct answer, which is not always guaranteed by heuristic algorithms. The QAOA ansatz is derived from the trotterization of the adiabatic evolution, allowing to reach global optimum with a sufficiently large numbers of layers, at least with parameters chosen to be sufficiently small numbers, say $\frac{1}{l}$ for l layers. Unfortunately, expecting such a convergence with a small number (thus NISQ-friendly) of layers is not obvious. Nevertheless, it is shown that QAOA outperforms the best classical solver for a particular decision problem in a recent work [73], providing a positive argument for its application in certain scenarios. For completeness, we would like to point out a few studies on QAOA. In [74], the authors show the existence of instances for which quantum and simulated annealing have exponentially small probability to find the solution, while those instances are exactly solvable by QAOA. On the other hand, in another recent work [75], it is shown that QA outperforms QAOA for randomly generated Ising instance when executed on the currently available hardware. Taking these into account, it remains inconclusive whether QAOA is a viable candidate for the integer factorization problem.

Another concern raised is that solving integer factorization problem with QAOA requires a

necessary step of transforming the problem into a SAT instance, thereby concealing the original interpretation behind a new formulation. This becomes particularly evident when attempting to approach the higher-order binary optimization formulation (like the one presented in [24]) using quantum annealing – in that case a quadratization procedure is necessary, which at the cost of introducing many additional variables, transforms the original pseudo-Boolean polynomial into a *quadratic* pseudo-Boolean polynomial. However, this hinders exploiting the original formulation of the problem, reducing (as mentioned in [36] in the comments section) the original problem to a SAT instance, for which the complexity is unlikely to be better than $c^{\log m}$ for any $c > 1$. On the contrary, for QAOA, no-meaning variables like the bits coming from the quadratization are not necessary, as in the gate-based machines higher-order terms can be easily implemented. Finally, such bits will be definitely not present when formulating the objective function via the procedure for Func-QAOA.

Even though such bits do not exist in the model, there might be some other issues which 'strengthen' the SAT-opted optimization evolution. The first example is the choice of the X-mixer, in which the amplitude-transfer graph represents a hyper-cube, preferring single-bit change. In order to avoid this, an all-to-all mixer should be used or nearest in modulo numbers should be swapped. The first one can be easily implemented with Grover mixer [44], while some efforts towards the second one can be found in [76], where the authors use the fact the corresponding mixer is a circulant matrix. Both mixers contribute to moving away from the interpretation of the landscape as a SAT instance.

Finally, perhaps the most serious issue with QAOA and also FUNC-QAOA is the choice of appropriate certificates. In order to demonstrate any advantage over the best classical algorithm, QAOA has to run in subexponential time. However, if the success probability is $\mathcal{O}(1/c^n)$ for some $c > 1$, it is unlikely to expect a super-polynomial improvement. Note that in all the Func-QAOA examples provided earlier, like multiplication, modulo and order finding, the success probability was exponentially small, making them not so interesting candidates. Better candidates are likely to be found among the elliptic curve factorization methods. Note that Func-QAOA might prove to be particularly beneficial for such certificates, as it is easier to design complicated certificates using universality of the gate-based machines. However, the oracles for these algorithms require subexponentially many gates, which make them impractical for the NISQ-era. Despite our efforts, we have not been able to find any suitable certificate that would simultaneously require only a polynomial number of gates and qubits, while also having a subexponential gate count.

3.3   Search space reduction

The previously presented advantages of Func-QAOA mitigate the fundamental issues arising from using QAOA-like algorithms for integer factorization. In this section, we explore how the search space can be reduced further by means of careful state preparation and Grover Mixer choice. Originally, for QAOA with the X-mixer, all the solutions were considered as the initial state is the superposition of all states. However in Quantum Alternating Operator Ansatz [42],

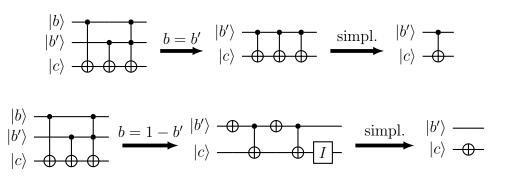Figure 2: Circuit simplification by bit substitution.

Figure 3: Circuit Simplification by bit redundancy.

and later in [43,44], alternative initial states are proposed, namely the superposition of one-hot vectors and the superposition of permutations. Reducing the search space size has a significant impact on the optimization quality, as it was numerically observed in [38]. In this section, as a pedagogical example, we will consider multiplication-based variant, however our results can be naturally considered for any other variant of the problem.

Let us first recall VQF [24]. In there, three different groups of bits are introduced: $\{p_i\}_i$ forming a register for the factor p, $\{q_j\}_j$ forming a register for the other factor q, and finally $\{z_k\}_k$ which are the carry-out bits coming from the grade-school multiplication of $\{p_i\}_i$ and $\{q_j\}_j$. The carry-outs bits are necessary to certify whether the product of p and q is equal to the biprime m. Since the problem is formulated as a Higher-Order Binary Optimization (HOBO) with X-mixer, z_k bring additional degrees of freedom, while not introducing any information about the problem: for any values of p and q, there is a unique natural assignment that can be obtained by taking the true carry-outs occurring in the multiplication. Recall that by using Func-QAOA, we were able to reduce the search space size from $\mathcal{O}\big(\log^2 m\big)$ into provable $\mathcal{O}(\log m)$ without any heuristic preprocessing of the problem. This was achieved by avoiding the carry bits, dropping the $\{q_j\}_j$ bits from the search space and optimizing only over the $\{p_i\}_i$ bits by choosing modulo certificate instead of multiplication.

While the above is simple and widely applicable reduction in Func-QAOA framework [38], one can save additional resources by carefully modifying the initial state. Note that in [24], the quadratic constraints are used to generate simplification rules, through which the original

HOBO can be simplified during the classical preprocessing. For the proposed QAOA, the preprocessing was used to remove some qubits and the corresponding gates. For Func-QAOA, we use the clauses generated in VQF to simplify the initial state. We consider three scenarios:

1. *Bit substitution*: clauses imply $b = 0$ or $b = 1$ for a binary variable b.

2. *Bit redundancy*: clauses imply $b = b'$ or $b = 1 - b'$ for binary variables b, b'.

3. *Superposition reduction*: clauses imply binary variables $b_1, \ldots, b_k$ can attain only particular values.

For each case, we choose a unique strategy for simplifying the circuit. The simplest case is the bit substitution, where the bits provide no information from the context of the optimization and should be set to $|0\rangle$ or $|1\rangle$ for the whole process. However, this means that those qubits can be just removed from the circuit, and all the gates applied afterwards can be simplified accordingly, as presented in Fig. 2. As an example, for the multiplication, it is reasonable to assume that the least significant bits of the potential factors are 1 – otherwise the biprime would need to be even, which is unrealistic for applications in cryptography.

For the bit redundancy, one can start with the Bell states $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ for constraints $b = b'$ and $b = 1 - b'$ respectively. This minor modification requires only a tiny increase in the number of gates, one CNOT for the state preparation and two for the Grover mixer. However, it effectively halves the size of the search space. In the case one of the bits serves solely as a control, the bit can be removed from the circuit and the gates where b acts as a control can be implemented as follows. W.l.o.g., suppose we remove b'. If $b = b'$, then the gates controlled on b' can be controlled by b. If $b = 1 - b'$, first a NOT gate should be applied on b, then the gate is applied with a control on b and NOT gate should applied again to revert b back to its original state. Through this simplification, the search space size is halved and one qubit is saved. Those examples are visualized in Fig. 3.

While very effective, we do not expect the above simplifications to appear too frequently. On the contrary, it is much more likely that more complicated clause simplifications can be found, in which for a subset of bits only some possible assignments are valid. Here, a dedicated method is to create a superposition of only those computational basis states that satisfy the clause. If the number of qubits over which the clause is defined is sufficiently small, say at most $c \log(n)$, one can use the brute-force method to determine all bit assignments satisfying the clause and then prepare a superposition using a general state preparation method [77, 78] using $\mathcal{O}(n^c)$ gates.

Based on the VQF formulation, we show how the search space can be reduced for $m = 77$, which is demonstrated in Fig. 4. First we identify two clauses with non-overlapping variables p_i and q_j; $p_6 q_3 = 0$ and $p_2 + 2q_1 + q_2 - 2z_{23} = 1$. Then we identify all the valid assignments for these bits. Note that for the second clause, we look for an assignment of p_2, q_1, q_2 so that there *exists* a value for z_{23} and the clause is satisfied. The valid assignments are presented in Fig. 4. Then, quantum circuits are generated which will produce a uniform superposition
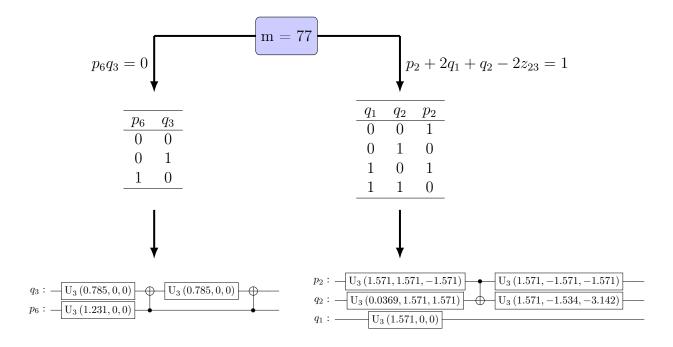
Figure 4: The search space reduction for $m = 77$ by superposition reduction.

of valid assignments for the corresponding qubits, for instance $\frac{1}{\sqrt{3}}(|00\rangle + |10\rangle + |01\rangle)$ for the variables p_6, q_3. The chosen clauses allow reducing the number of possible solutions from 4 to 3 and from 8 to 4, respectively. Consequently, the first simplification removes 25% of the search space, while the second one removes 50% of the search space.

Note that removing even a minimal number of bits may have a significant effect on the reduction of the effective space size. To see this, let us assume an optimistic scenario with n qubits and n/c clauses, each defined on a fixed, c independent bits. Suppose $b < 2^c$ bits are removed from each clause. Then the new search space size is

$$\log_2 \prod_{i=1}^{n/c} (2^c - b) = n \left(1 + \frac{1}{c} \log_2 \left(1 - \frac{b}{2^c}\right)\right), \tag{3}$$

whereas the original search space size is n. We can see that for any fixed b and c, we are reducing the effective space size by a multiplicative constant. As an example, if we take $b = \frac{1}{2}2^c$, we have the effective space size $n(1 - \frac{1}{c})$. Note that being able to find n/c clauses on non-overlapping variables might in general be a challenging task, e.g. in VQF formulation most of the clauses act on most of the binary variables instead of a tiny subset of them.

In addition, we considered how much the search space can be reduced for clauses with a particular number of p_i and q_j variables. The results are presented in Fig. 5. We generated clauses using the classical preprocessing rules as proposed in [24] and we identified those for which the total number of p_i and q_j variables appearing in a clause was at most 16. We can see
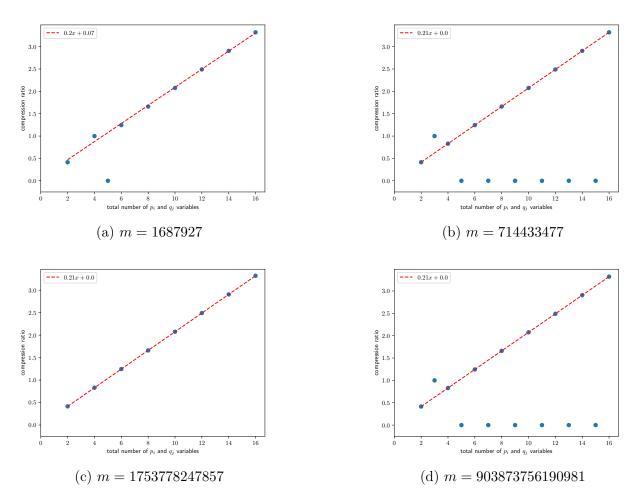
16

(a) $m = 1687927$

(b) $m = 714433477$

(c) $m = 1753778247857$

(d) $m = 903873756190981$

Figure 5: Compression ratios for different total numbers of p_i and q_j variables in clauses.

that the compression ratio, defined as the logarithm of the number of all assignments minus number of invalid assignments over p_i, q_j grows at most linearly with the total number of p_i and q_j variables, however it does not change with length of the biprime. This is because the clauses with chosen number of variables are mostly those specifying conditions for the least and the most significant bits of the product, and they often have a similar form.

Note that the origin of finding such clauses does not have to be directly connected to the certificate used. In particular all the rules generated for the smaller prime based on the integer formulation for multiplication can be also used for the modulo-based certification. Note that the simplification can be done only based on some of the constraints, and the full formulation does not need to be constructed.

It is important to acknowledge that this simplification is unlikely to reduce the search space to $\mathcal{O}(\log n)$ qubits. These kinds of simplifications are classical in nature and one would expect the same simplification to occur if the same problem is tackled using a classical algorithm e.g. simulated annealing, which would make the problem classically tractable.

4   Discussion and Conclusion

In this paper, we pointed some potential challenges that may arise when attempting to solve the factorization problem with QAOA. This includes in particular representing factorization problem as a SAT instance, involving bits that are redundant or even lack context which occur as a result of formulating the problem instance as a QUBO or HOBO, and using a bit-oriented mixer. We showed, how those limitations can be potentially overcame by leveraging the results from the current literature, with a particular focus on Func-QAOA, a variant of QAOA which relies on the universality of the gate-based model. With Func-QAOA, not only the context of the problem remains visible in the QAOA ansatz, but it also allows a natural incorporation of more complicated certifications for the problem.

While our work introduces a potentially interesting framework for integer factorization using QAOA, it remains unclear whether a practical certificate exists within the limitations of the NISQ era. Among all the certificates we provided and are aware of, the ones that offer subexponential speed-up for classical computing require subexponential time and qubits for certification. This clearly goes beyond the capabilities of current devices, as achieving sufficiently faithful evolution with subexponentially many gates would likely require fidelity that would allow for running Shor's algorithm. Therefore, we leave an open question whether there is a certificate which has subexponential probability of finding a correct solution, with a polynomial number of gates. While one might hope that exploring the search space could eliminate the requirement for subexponential success probability, this would imply that QAOA achieves faster-than-polynomial speedup for that optimization problem, which we believe to be highly unlikely.

We would also like to emphasize that the challenge of finding a good certificate is not the only aspect that needs to be addressed. Despite the promising results on applying QAOA for

a *decision problem K*-SAT [73], it is far from evident whether similar results can be expected for factorization. Furthermore, there are crucial factors that have not been taken into account, but will contribute to a big constant, and likely at least polynomial multiplicative overhead for the time of finding the correct solution. These factors include the limited connectivity of the quantum devices, the impact of noise and decoherence, the cost of estimating the energy of the quantum state, and the cost of the optimization process itself. Taking all of those into account, and the unpromising recent results for QAOA on real quantum machines [75], we assert that, at the very least, in the near future, QAOA is unlikely to pose a threat to the security of our bank accounts.

Acknowledgements

Additional information

Code used for generating Figs. 4 and 5 is available at `https://doi.org/10.5281/zenodo.8363223`.

References

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[2] R. S. Lehman, "Factoring large integers," *Mathematics of Computation*, vol. 28, no. 126, pp. 637–646, 1974.

[3] J. M. Pollard, "A Monte Carlo method for factorization," *BIT Numerical Mathematics*, vol. 15, pp. 331–334, 1975.

[4] H. W. Lenstra Jr, "Factoring integers with elliptic curves," *Annals of Mathematics*, vol. 126, pp. 649–673, 1987.

[5] J. P. Buhler, H. W. Lenstra, and C. Pomerance, "Factoring integers with the number field sieve," in *The Development of the Number Field Sieve*, pp. 50–94, Springer, 1993.

[6] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994.

[7] "Winners of the 2023 breakthrough prizes in life sciences, mathematics and fundamental physics announced." `https://breakthroughprize.org/News/73`, 2022.

[8] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, no. 6866, pp. 883–887, 2001.

[9] C.-Y. Lu, D. E. Browne, T. Yang, and J.-W. Pan, "Demonstration of a compiled version of Shor's quantum factoring algorithm using photonic qubits," *Physical Review Letters*, vol. 99, no. 25, p. 250504, 2007.

[10] A. Politi, J. C. Matthews, and J. L. O'Brien, "Shor's quantum factoring algorithm on a photonic chip," *Science*, vol. 325, no. 5945, pp. 1221–1221, 2009.

[11] S. Beauregard, "Circuit for Shor's algorithm using $2n + 3$ qubits," *Quantum Information & Computation*, vol. 3, no. 2, p. 175–185, 2003.

[12] Y. Takahashi and N. Kunihiro, "A quantum circuit for Shor's factoring algorithm using 2n+ 2 qubits," *Quantum Information & Computation*, vol. 6, no. 2, pp. 184–192, 2006.

[13] T. Häner, M. Roetteler, and K. M. Svore, "Factoring using 2n + 2 qubits with Toffoli based modular multiplication," *Quantum Information & Computation*, vol. 17, no. 7–8, p. 673–684, 2017.

[14] E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien, "Experimental realization of Shor's quantum factoring algorithm using qubit recycling," *Nature Photonics*, vol. 6, no. 11, pp. 773–776, 2012.

[15] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[16] M. Mosca, "Cybersecurity in an era with quantum computers: will we be ready?," *IEEE Security & Privacy*, vol. 16, no. 5, pp. 38–41, 2018.

[17] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum computation by adiabatic evolution," *arXiv preprint quant-ph/0001106*, 2000.

[18] X. Peng, Z. Liao, N. Xu, G. Qin, X. Zhou, D. Suter, and J. Du, "Quantum adiabatic algorithm for factorization and its experimental implementation," *Physical Review Letters*, vol. 101, no. 22, p. 220405, 2008.

[19] G. Schaller and R. Schützhold, "The role of symmetries in adiabatic quantum algorithms," *Quantum Information & Computation*, vol. 10, no. 1, p. 109–140, 2010.

[20] N. Xu, J. Zhu, D. Lu, X. Zhou, X. Peng, and J. Du, "Quantum factorization of 143 on a dipolar-coupling nuclear magnetic resonance system," *Physical Review Letters*, vol. 108, no. 13, p. 130501, 2012.

[21] N. S. Dattani and N. Bryans, "Quantum factorization of 56153 with only 4 qubits," *arXiv preprint arXiv:1411.6758*, 2014.

[22] W. Peng, B. Wang, F. Hu, Y. Wang, X. Fang, X. Chen, and C. Wang, "Factoring larger integers with fewer qubits via quantum annealing with optimized parameters," *SCIENCE CHINA Physics, Mechanics & Astronomy*, vol. 62, no. 6, pp. 1–8, 2019.

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

[24] E. Anschuetz, J. Olson, A. Aspuru-Guzik, and Y. Cao, "Variational quantum factoring," in *Proceedings of the International Workshop on Quantum Technology and Optimization Problems*, pp. 74–85, Springer, 2019.

[25] A. H. Karamlou, W. A. Simon, A. Katabarwa, T. L. Scholten, B. Peropadre, and Y. Cao, "Analyzing the performance of variational quantum factoring on a superconducting quantum processor," *npj Quantum Information*, vol. 7, no. 1, pp. 1–6, 2021.

[26] B. Yan, Z. Tan, S. Wei, H. Jiang, W. Wang, H. Wang, L. Luo, Q. Duan, Y. Liu, W. Shi, *et al.*, "Factoring integers with sublinear resources on a superconducting quantum processor," *arXiv preprint arXiv:2212.12372*, 2022.

[27] C. J. Burges, "Factoring as optimization," *Microsoft Research MSR-TR-200*, 2002.

[28] S. Jiang, K. A. Britt, A. J. McCaskey, T. S. Humble, and S. Kais, "Quantum annealing for prime factorization," *Scientific Reports*, vol. 8, no. 1, p. 17667, 2018.

[29] B. Wang, F. Hu, H. Yao, and C. Wang, "Prime factorization algorithm based on parameter optimization of Ising model," *Scientific Reports*, vol. 10, no. 1, p. 7106, 2020.

[30] B. Apolloni, C. Carvalho, and D. De Falco, "Quantum stochastic optimization," *Stochastic Processes and their Applications*, vol. 33, no. 2, pp. 233–244, 1989.

[31] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Physical Review E*, vol. 58, no. 5, p. 5355, 1998.

[32] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[33] K. Jun and H. Lee, "HUBO and QUBO models for prime factorization," *Scientific Reports*, vol. 13, no. 1, p. 10080, 2023.

[34] fgrieu, "Largest integer factored by Shor's algorithm?." `https://crypto.stackexchange.com/questions/59795/largest-integer-factored-by-shors-algorithm/59796#59796`.

[35] C. Gidney, "Factoring the largest number ever with a quantum computer." `https://algassert.com/post/2000`, 2020.

[36] S. Aaronson, "Shtetl-optimized: Quantum computing motte-and-baileys." `https://scottaaronson.blog/?p=4447/`, Dec 2019.

[37] S. Aaronson, "Shtetl-optimized: Cargo cult quantum factoring." `https://scottaaronson.blog/?p=6957`, Jan 2023.

[38] B. Bakó, A. Glos, Ö. Salehi, and Z. Zimborás, "Near-optimal circuit design for variational quantum optimization," *arXiv preprint arXiv:2209.03386*, 2022.

[39] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM symposium on Theory of Computing*, pp. 212–219, 1996.

[40] D. J. Bernstein, N. Heninger, P. Lou, and L. Valenta, "Post-quantum RSA," in *Proceedings of the International Workshop on Post-Quantum Cryptography*, pp. 311–329, Springer, 2017.

[41] M. Born and V. Fock, "Beweis des adiabatensatzes," *Zeitschrift für Physik*, vol. 51, no. 3, pp. 165–180, 1928.

[42] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, 2019.

[43] Z. Wang, N. C. Rubin, J. M. Dominy, and E. G. Rieffel, "XY mixers: Analytical and numerical results for the quantum alternating operator ansatz," *Physical Review A*, vol. 101, no. 1, p. 012320, 2020.

[44] A. Bärtschi and S. Eidenbenz, "Grover mixers for QAOA: Shifting complexity from mixer design to state preparation," in *Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 72–82, IEEE, 2020.

[45] N. P. Sawaya, A. T. Schmitz, and S. Hadfield, "Encoding trade-offs and design toolkits in quantum algorithms for discrete optimization: Coloring, routing, scheduling, and other problems," *arXiv preprint arXiv:2203.14432*, 2022.

[46] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, p. 5, 2014.

[47] A. Glos, A. Krawiec, and Z. Zimborás, "Space-efficient binary optimization for variational quantum computing," *npj Quantum Information*, vol. 8, no. 1, pp. 1–8, 2022.

[48] 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 efficient embedding," in *Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 56–62, IEEE, 2020.

[49] P. D. de la Grand'rive and J.-F. Hullo, "Knapsack problem variants of QAOA for battery revenue optimisation," *arXiv preprint arXiv:1908.02210*, 2019.

[50] F. G. Fuchs, H. Ø. Kolden, N. H. Aase, and G. Sartor, "Efficient encoding of the weighted MAX-K-CUT on a quantum computer using QAOA," *SN Computer Science*, vol. 2, no. 2, pp. 1–14, 2021.

[51] R. Rines and I. Chuang, "High performance quantum modular multipliers," *arXiv preprint arXiv:1801.01081*, 2018.

[52] C. Gidney, "Asymptotically efficient quantum Karatsuba multiplication," *arXiv preprint arXiv:1904.07356*, 2019.

[53] J. Álvarez Sánchez, J. Álvarez Bravo, and L. Nieto, "A quantum architecture for multiplying signed integers," *Journal of Physics: Conference Series*, vol. 128, p. 012013, 10 2008.

[54] S. Dutta, D. Bhattacharjee, and A. Chattopadhyay, "Quantum circuits for Toom-Cook multiplication," *Physical Review A*, vol. 98, no. 1, p. 012311, 2018.

[55] H. T. Larasati, A. M. Awaludin, J. Ji, and H. Kim, "Quantum circuit design of Toom 3-way multiplication," *Applied Sciences*, vol. 11, p. 3752, Apr 2021.

[56] H. M. H. Babu, "Cost-efficient design of a quantum multiplier-accumulator unit," *Quantum Information Processing*, vol. 16, p. 30, Jan. 2017.

[57] E. Şahin, "Quantum arithmetic operations based on quantum Fourier transform on signed integers," *International Journal of Quantum Information*, vol. 18, no. 06, p. 2050035, 2020.

[58] H. Thapliyal, E. Muñoz-Coreas, T. S. S. Varun, and T. S. Humble, "Quantum circuit designs of integer division optimizing T-count and T-depth," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 1045–1056, 2021.

[59] L. Jamal and H. M. H. Babu, "Efficient approaches to design a reversible floating point divider," in *Proceedings of the 2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 3004–3007, 2013.

[60] T. G. Draper, "Addition on a quantum computer," *arXiv preprint quant-ph/0008033*, 2000.

[61] L. Ruiz-Perez and J. C. Garcia-Escartin, "Quantum arithmetic with the quantum Fourier transform," *Quantum Information Processing*, vol. 16, no. 6, 2017.

[62] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," *arXiv preprint quant-ph/0410184*, 2004.

[63] F. Wang, M. Luo, H.-R. Li, Z. Qu, and X. Wang, "Improved quantum ripple-carry addition circuit," *Science China Information Sciences*, vol. 59, pp. 1–8, 2016.

[64] H. Thapliyal and N. Ranganathan, "Design of efficient reversible logic-based binary and BCD adder circuits," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 9, no. 3, pp. 1–31, 2013.

[65] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," *Quantum Information & Computation*, vol. 6, no. 4, p. 351–369, 2006.

[66] H. Thapliyal, H. V. Jayashree, A. N. Nagamani, and H. R. Arabnia, "Progress in reversible processor design: A novel methodology for reversible carry look-ahead adder," in *Transactions on Computational Science XVII* (M. L. Gavrilova and C. J. K. Tan, eds.), pp. 73–97, Springer, 2013.

[67] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," *Quantum Information & Computation*, vol. 10, no. 9, p. 872–890, 2010.

[68] Y. Takahashi and N. Kunihiro, "A fast quantum circuit for addition with few qubits," *Quantum Information & Computation*, vol. 8, no. 6, pp. 636–649, 2008.

[69] C. Gidney, "Quantum block lookahead adders and the wait for magic states," *arXiv preprint arXiv:2012.01624*, 2020.

[70] B.-S. Choi and R. V. Meter, "A $\Theta(\sqrt{n})$-depth quantum adder on the 2D NTC quantum computer architecture," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 8, no. 3, pp. 1–22, 2012.

[71] H. Thapliyal, "Mapping of subtractor and adder-subtractor circuits on reversible quantum gates," in *Transactions on Computational Science XXVII* (M. L. Gavrilova and C. K. Tan, eds.), pp. 10–34, Springer, 2016.

[72] H. Thapliyal and N. Ranganathan, "A new design of the reversible subtractor circuit," in *Proceedings of the 2011 11th IEEE International Conference on Nanotechnology*, pp. 1430–1435, IEEE, 2011.

[73] S. Boulebnane and A. Montanaro, "Solving boolean satisfiability problems with the quantum approximate optimization algorithm," *arXiv preprint arXiv:2208.06909*, 2022.

[74] M. Streif and M. Leib, "Comparison of QAOA with quantum and simulated annealing," *arXiv preprint arXiv:1901.01903*, 2019.

[75] E. Pelofske, A. Bärtschi, and S. Eidenbenz, "Quantum annealing vs. QAOA: 127 qubit higher-order Ising problems on NISQ computers," in *Proceedings of the International Conference on High Performance Computing*, pp. 240–258, Springer, 2023.

[76] X. Qiang, T. Loke, A. Montanaro, K. Aungskunsiri, X. Zhou, J. L. O'Brien, J. B. Wang, and J. C. Matthews, "Efficient quantum walk on a quantum processor," *Nature Communications*, vol. 7, no. 1, p. 11511, 2016.

[77] V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum logic circuits," in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, pp. 272–275, 2005.

[78] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, "Transformation of quantum states using uniformly controlled rotations," *Quantum Information & Computation*, vol. 5, no. 6, pp. 467–473, 2005.