Hybrid algorithm for integer factorization on quantum computers

Inventors: Eric R. Anschuetz, Yudong Cao, Jonathan Olson

Integer factorization is one of the first practically relevant problems that can be solved exponentially faster on a quantum computer than any currently known methods for classical computation using the famed Shor's algorithm [1]. The ability to efficiently solve integer factorization has deep consequences for cryptography. In particular, encryption schemes based on Abelian groups such as RSA and elliptic curves can be entirely compromised if efficient factorization were feasible. However, Shor's algorithm requires quantum computers to perform sequences of quantum operations that are too lengthy for noisy intermediate-scale quantum devices that are available in the near-term. This renders the potential of quantum computers compromising Abelian group based encryptions a distant reality.

Here, we describe a scheme for integer factorization on near-term quantum computers. We summarize the workflow in Figure 1. The overall setup of the algorithm involves maintaining and updating a set of solution candidates, while checking whether any of the candidates give rise to valid factorization of the input (Figure 1b). The algorithm terminates once a satisfying solution has been found. In particular, to account for the fact that near-term quantum devices may not have enough qubits to accommodate the entire factoring problem, we use a hybrid quantum-classical approach to generate solution candidates (Figure 1b). To produce an n-bit candidate with n possibly greater than the number of qubits available on the quantum computer, we first use the classical computer to assign m bits, giving rise to a new Hamiltonian acting on considerably fewer qubits than n which can be fit onto the quantum computer. Then, we use a quantum computer to find an optimized state for the new Hamiltonian, which assigns values to the remaining bits in the n-bit string. The completed string is then fed into another classical solver B which maintains and updates solution candidates. Then the solution candidate(s) is checked to see if it gives rise to the correct factors. If it does then we can glean factors p and q from the solution. Otherwise information about the solution candidate is fed back into solver A to guide its action in the next iteration.

Examples of potential classical solvers A include a classical Boolean equation reducer

and a PromiseBall solver, both of which are discussed in detail in Sec. III. Possible realizations of classical solvers B include simulated annealing and genetic algorithms, which maintain and update n-bit strings to produce improved batches of solution candidates. On near-term devices, the quantum solver can—for instance—be realized using the quantum approximate optimization algorithm [2] or the hardware-efficient variational quantum eigensolver [3], which have proven effective in solving ground states of Ising Hamiltonians and (by equivalence reduction) a few problems that are NP-complete. We discuss these quantum solvers in greater detail in Sec. II.

I. ENCODING FACTORING INTO AN ISING HAMILTONIAN

It has been previously noticed that factoring can be cast as the minimization of a cost function [4], which can then be encoded into the ground state of an Ising Hamiltonian [5–7]. To see this, consider the factoring of

$$m = \sum_{i=0}^{n_m} 2^i m_i \tag{1}$$

into

$$p = \sum_{i=0}^{n_p} 2^i p_i \tag{2}$$

and

$$q = \sum_{i=0}^{n_q} 2^i q_i, \tag{3}$$

where n_a is the number of bits of a, and $\{a_j\}$ form the bit representation of a. We use $\{b_k\}$ to denote the collection of all bit variables. When n_p and n_q are unknown (as they are *a priori* when only given a number m to factor), one may set $n_p = n_m$ and $n_q = \left\lceil \frac{n_m}{2} \right\rceil$ [4]. As $m = pq$, this bit expansion yields the set of clauses

$$C_i = \sum_{j=0}^{n_q} q_j p_{i-j} + \sum_{j=0}^{i} z_{j,i} - m_i - \sum_{j=1}^{\ell_i} 2^j z_{i,i+j} \tag{4}$$

where $0 \leq i \leq n_m$ [4]. Here, $z_{i,j}$ represents the carry bit from bit position i into bit position j and

$$\ell_i = \lceil \lg(\mu_i) \rceil, \tag{5}$$

where μ_i is the value of the first three terms of (4) if all unknown variables are set to one (i.e., the maximum possible value of the first three terms of (4)) [5, 7]. All $z_{i,i+j}$ for $j > \ell_i$

are set to zero, as if they were one then the subtrahend of (4) would be so large as to make satisfying C_i impossible. Thus, factoring can be represented as solving the Boolean satisfaction problem over

$$C = \sum_{i=0}^{n_m} C_i. \tag{6}$$

This corresponds to the minimization of the classical energy function

$$E = \sum_{i=0}^{n_m} C_i^2, \tag{7}$$

which has a natural quantum representation as a *factoring Hamiltonian*

$$H = \sum_{i=0}^{n_m} \hat{C}_i^2 \tag{8}$$

where each bit variable in the energy function is now quantized by taking the transformation

$$b_k \to \frac{1}{2}\left(1 - \sigma_k^z\right). \tag{9}$$

We further note that H can be represented in quadratic form by substituting each product $q_j p_{i-j}$ with a new binary variable $w_{i,j}$ and adding additional constraints to the Hamiltonian [5]. Therefore, there is a natural encoding of a factoring instance into the ground state of an Ising Hamiltonian. If the number being factored is a biprime, analogous clauses to (4) that must not be satisfied can be derived by expanding $M = pq \prod_{i=1}^{m_c} c^{(i)}$ for m_c extraneous factors. These clauses can then be added as an energy penalty to (7) to raise the energy of some non-satisfying assignments and aid later steps in our factorization scheme in finding the ground state of the factoring Hamiltonian.

II. SOLVING ISING HAMILTONIANS ON NEAR-TERM QUANTUM DEVICES

Our technical innovation is using a near-term approximate quantum ground state solver as a means to approximately factor numbers on near-term gate model quantum computers. This is accomplished by using such a quantum solver to find the ground state of the factoring Hamiltonian given by (8). Here, we give some examples of near-term quantum algorithms that could act as the quantum solver in this framework.

A. Quantum Approximate Optimization Algorithm

The *quantum approximate optimization algorithm* (QAOA) is a hybrid classical/quantum algorithm for near-term quantum computers that approximately solves classical optimization problems [2]. The goal of the algorithm is to satisfy (i.e., find the simultaneous zeros of) clauses C_i over n variables, which is phrased as the minimization of a classical *cost Hamiltonian*

$$H_c = \sum_i C_i^2. \tag{10}$$

This is done by preparing an ansatz state

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = \prod_{i=1}^{s} \left(\exp\left(-i\beta_i H_a\right) \exp\left(-i\gamma_i H_c\right)\right) |+\rangle^{\otimes n} \tag{11}$$

parametrized by angles $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ over n qubits. Here, H_a is the *admixing Hamiltonian*

$$H_a = \sum_{i=1}^{n} \sigma_i^x. \tag{12}$$

For a fixed s, QAOA uses a classical optimizer to minimize the cost function

$$M\left(\boldsymbol{\beta}, \boldsymbol{\gamma}\right) = \langle \boldsymbol{\beta}, \boldsymbol{\gamma}| \, H_c \, |\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle \,, \tag{13}$$

m whose global minima are given by computational basis states that satisfy the maximum number of clauses.

B. Hardware-Efficient Variational Quantum Eigensolver

The *hardware-efficient variational quantum eigensolver* (HEVQE) is a hybrid quantum-classical algorithm for near-term quantum computers that finds the approximate ground state of quantum Hamiltonians [3]. This is done by preparing the hardware-efficient ansatz

$$|\{\boldsymbol{\theta}_i\}_{i=0}^{s}\rangle = \prod_{i=1}^{s} \left(U_{\text{rot}}\left(\boldsymbol{\theta}_i\right) U_{\text{ent}}\right) U_{\text{rot}}\left(\boldsymbol{\theta}_0\right) |0\rangle^{\otimes n} \tag{14}$$

over n qubits. Here, $U_{\text{rot}}\left(\boldsymbol{\theta}_i\right)$ represents arbitrary single-qubit rotations parameterized by the Euler angles $\boldsymbol{\theta}_i$ and U_{ent} is an arbitrary entangling gate that in practice is taken to be an entangling gate that is implemented with high fidelity on the architecture the ansatz is

prepared on. Then, for a fixed s and problem Hamiltonian H_p, HEVQE uses a classical optimizer to minimize the cost function

$$M\left(\{\boldsymbol{\theta}_i\}_{i=0}^{s}\right) = \langle\{\boldsymbol{\theta}_i\}_{i=0}^{s}|\, H_p\, |\{\boldsymbol{\theta}_i\}_{i=0}^{s}\rangle\,, \tag{15}$$

such that $|\{\boldsymbol{\theta}_i\}_{i=0}^{s}\rangle$ approximates the ground state of H_p.

III. A CLASSICAL/QUANTUM HYBRID FRAMEWORK FOR FACTORING

Though in principle our application of a quantum solver to the factoring Hamiltonian would be enough to factor numbers in the $s \to \infty$ limit, in practice classical algorithms can be used to minimize the resource requirements of the quantum device. In our algorithm these classical algorithms are represented as a *classical presolver* A and a *classical postsolver* B (Figure 1b). The goal of the classical presolver is to directly solve the factoring problem on a subset of bit variables efficiently, directly lowering the resource requirements of the quantum solver. The goal of the classical postsolver is to process samples drawn from the output state of the quantum solver, reducing the precision requirements of the quantum solver. For example, samples drawn from the output state of the quantum solver could seed a classical genetic algorithm or classical simulated annealing algorithm to optimize the classical energy function (7). In the rest of this section, we give detailed examples of potential classical presolvers.

A. Classical Boolean Equation Reduction

One method for reducing the resource requirements of the quantum solver is to directly solve for a subset of the bit variables in the optimization problem that are easy to solve for classically [6, 7]. This reduction iterates through all clauses C_i as given by (4). In the following discussion, let $x, y \in \mathbb{F}_2$ be unknown variables, and $a, b \in \mathbb{Z}^+$ known constants (as they are either functions of m or are known through the classical preprocessing of C_j for

$j < i$). For a given C_i, we apply the classical preprocessing rules:

$$\begin{aligned} C_i = xy - 1 &\implies x = y = 1, \\ C_i = x + y - 2 &\implies x = y = 1, \\ C_i = x + y - 1 &\implies xy = 0, \\ C_i = x + a - by &\implies y = 1, \\ C_i = x - y &\implies x = y. \end{aligned} \tag{16}$$

This classical preprocessing iterates through each of $O(n_m)$ terms in each of $O(n_m)$ clauses, costing a runtime of $O(n_m^2)$. However, in practice this greatly reduces the number of qubits needed for the quantum solver portion of the algorithm.

B. Reduction to PromiseBall

Another strategy for classical presolving is using a classical algorithm that calls the quantum solver on subsets of the problem. This was done in [8], where a quantum amplitude amplification algorithm is used to solve PromiseBall. PromiseBall asks to solve a k-SAT instance F given an assignment α and a guarantee that a satisfying assignment of F exists within Hamming distance r of α. As factoring is in NP and k-SAT is NP-complete for $k \geq 3$, all factoring problems can be phrased as PromiseBall instances. As there exist classical algorithms that solve PromiseBall that are recursive in r [9, 10], these classical presolvers are able to call quantum subroutines when the problem instance is small enough to fit on the quantum device. As amplitude amplification is not believed to be able to run on near-term quantum devices [11], the quantum subroutine could instead be substituted by a quantum solver to generate states which are then optimized over to find solutions to PromiseBall subinstances.

For instance, given r, one could use the hardware-efficient ansatz (14) as a quantum solver with this classical presolver. Using the hardware-efficient ansatz, one generates parameterized quantum states over $r^* = r \lg(k)$ qubits from which classical bit strings over r^* bits can be sampled from. These r^* bits can then be interpreted as r k-dits (that is, as r variables each over $\mathbb{Z}_k$). As in [9, 10], this r k-dit string then specifies a choice of up to r bits of α to flip, yielding α^*. Therefore, a classical optimization routine optimizes the ansatz over the cost function $F(\alpha^*)$, and as done in [8] this subroutine could be used in conjunction

with [9, 10] to yield a hybrid quantum-classical algorithm that solves PromiseBall and, therefore, factoring.

---

[1] P. W. Shor, SIAM Review **41**, 303 (1999), arXiv:9508027.

[2] E. Farhi, J. Goldstone, and S. Gutmann, (2014), arXiv:1411.4028.

[3] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Nature **549**, 242 (2017), arXiv:1704.05018.

[4] C. J. C. Burges, *Factoring as Optimization*, Tech. Rep. (2002).

[5] R. Dridi and H. Alghassi, Scientific Reports **7**, 43048 (2017), arXiv:1604.05796.

[6] N. S. Dattani and N. Bryans, (2014), arXiv:1411.6758.

[7] N. Xu, J. Zhu, D. Lu, X. Zhou, X. Peng, and J. Du, Physical Review Letters **108**, 130501 (2012), arXiv:1111.3726.

[8] V. Dunjko, Y. Ge, and J. I. Cirac, (2018), arXiv:1807.08970.

[9] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning, Theoretical Computer Science **289**, 69 (2002).

[10] R. A. Moser and D. Scheder, in *Proceedings of the 43rd annual ACM symposium on Theory of computing - STOC '11* (ACM Press, New York, New York, USA, 2011) p. 245.

[11] J. Preskill, Quantum **2**, 79 (2018), arXiv:1801.00862.

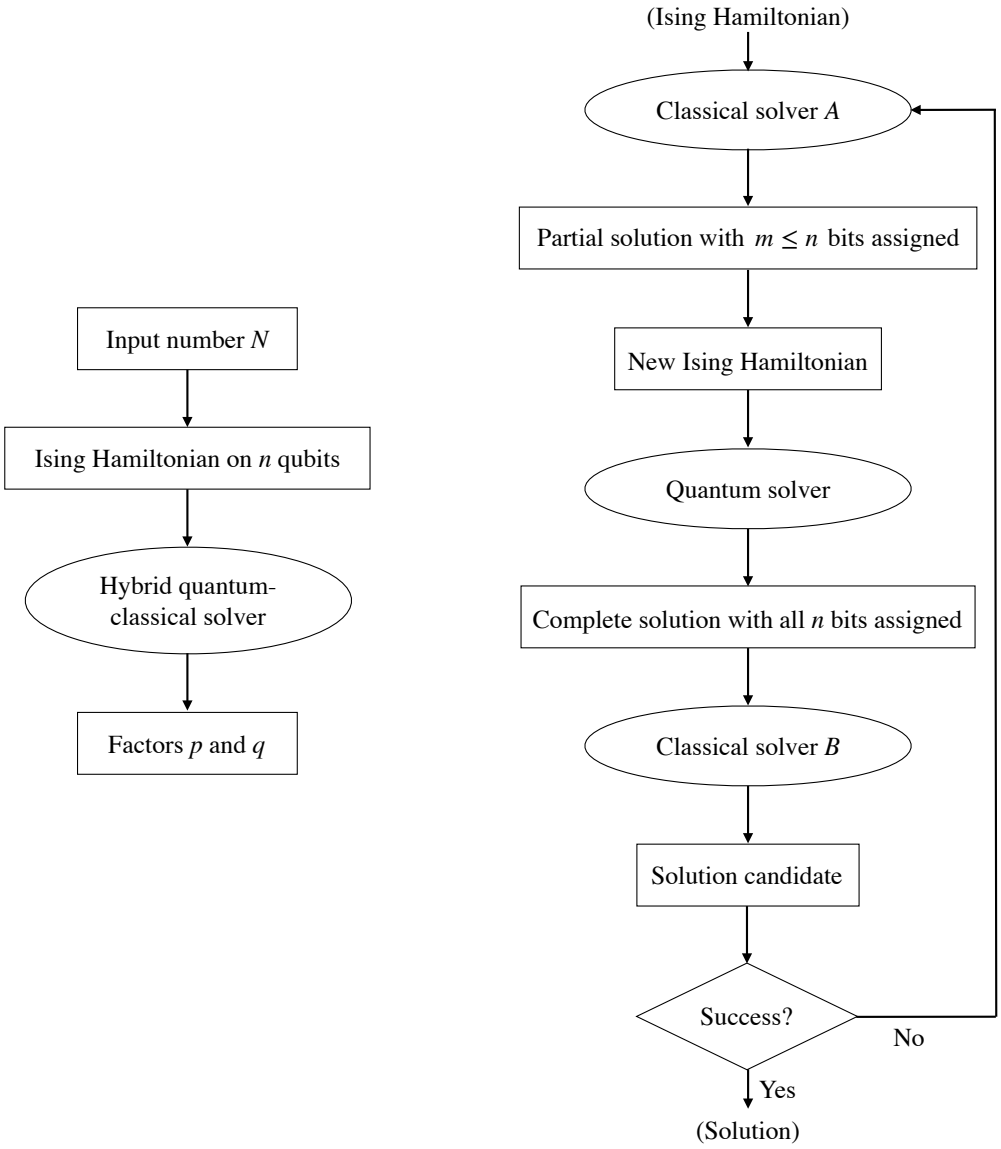

FIG. 1. Flow diagrams representing the workflow of the algorithm. Here we use circles to represent algorithmic steps and boxes to represent mathematical objects. (a) Main workflow of the hybrid scheme for factoring. The mapping from the input number to Ising Hamiltonian is described in Section I. (b) Workflow of the hybrid quantum-classical solver. The details of the hybrid quantum-classical solver are presented in Section III. One implementation of the quantum solver is presented in Section II.