


I, Mridul Gupta, declare that this MS project report titled "Semiprime factorization using AQC" contains results of my own research carried out under the guidance of Prof. Prasanta K. Panigrahi, IISER Kolkata, and to the best of my knowledge, it contains no materials previously published or written by any other person, or substantial proportions of material which has formed the basis of award of any other degree or diploma at IISER Kolkata or any other educational institution, except where due acknowledgement is made in the thesis.

Signed:

A handwritten signature in black ink, appearing to be 'Mridul Gupta', is written over a horizontal line.

Mridul Gupta  
Department of Physical Sciences  
IISER Kolkata  
Date: May 17, 2023

This is to certify that the MS project report entitled "Semiprime factorization using AQC" is a bonafide record of work done by Mridul Gupta, 5th year Integrated BS-MS student, Department of Physical Sciences, under my supervision during August 2022 - July 2023 in partial fulfillment of the requirements for the award of Integrated BS-MS in the Department of Physical Sciences at Indian Institute of Science Education and Research, Kolkata.

Signed:

*Prasanta K. Panigrahi*

---

**Prof. Prasanta K. Panigrahi**  
Department of Physical Sciences  
IISER Kolkata  
West Bengal 741246  
India

Date : May 16, 2023

# MS Project Report

## Semiprime Factorization using AQC

Mridul Gupta (18MS054)\*

Supervisor: Prof. Prasanta K. Panigrahi

May 17, 2023

### Introduction

Adiabatic Quantum Computation (AQC), also known as Quantum Annealing, is a computational model that uses quantum mechanics to find the lowest energy state of a cost Hamiltonian. [1] This model is used by D-Wave quantum computers and has been shown to be equivalent to the gate-circuit model used by IBM-Q computers. [2]

AQC is similar to a group of classical algorithms called Simulated Annealing (SA), which use statistical mechanics to solve optimization problems and other combinatorial search problems. AQC generally performs better than SA, taking less time to complete and being more accurate in edge cases (taking the help of quantum tunneling to escape out of local optimal solutions). One key difference between SA and AQC is that SA uses temperature as a control parameter to find the optimal configuration when temperature reaches zero, while the system in AQC is always close to the instantaneous ground state.

AQC started as an approach to solving optimization problems and has evolved into an important universal alternative to the standard circuit model of quantum computing, with deep connections to both classical and quantum

---

\*emgi@live.in

complexity theory and condensed matter physics. The time complexity for an adiabatic algorithm is dependent on the gap in the energy eigenvalues (spectral gap) of the Hamiltonian.

In this project we will implement an AQC algorithm for semiprime factorization problem and attempt to figure out the time complexity of it.

## Semiprime factorization problem

A semiprime number is a composite number that can be written as a product of exactly two prime numbers. These factors may be equal to each other. Semiprime numbers are used extensively in cryptography. These systems rely on the fact that finding large prime numbers and multiplying them is easy but finding the factor from the semiprime number is hard.

It is in fact intractable, which means that the time required to factorize a number is necessarily exponential in its length, or in other words has exponential time complexity. Semi-prime factorization is an increasingly important number theoretic problem, since it is computationally intractable. Further, this property has been applied in public-key cryptography, such as the Rivest–Shamir–Adleman (RSA) cryptosystem for secure digital communications.

Overall, semiprime numbers remain an active area of research with implications to cryptography and number theory.

## Overall Process

The overall process of finding the factors of a number  $N$  of length  $b_N$  can be understood using Flowchart 1.

Starting with our number  $N$ , we first choose our guess  $b_P, b_Q$  such that

$$b_N = b_P + b_Q \text{ or } b_P + b_Q + 1$$

we will need to loop over all possible guesses. We start by generating clauses based on our initial guess. Then, we apply variable reduction rules to make the clauses as information-dense as possible. If the rules return an error due to impossible clauses, we reject this iteration of our guess and start over with a new one.

At this point, we have completed the classical preprocessing part of our method. We use the reduced forms of clauses to build Hamiltonians that

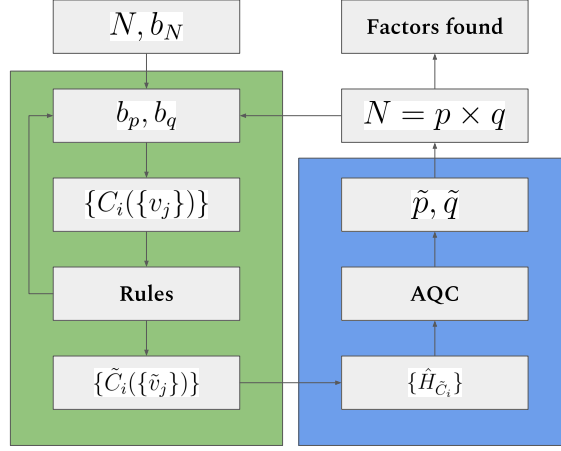


Figure 1: A Flowchart showing the overall workflow of factorizing semiprime  $N$  using AQC (blue) with some classical preprocessing (green)

assign the lowest energy to states encoding our solution. These Hamiltonians are used to create our problem Hamiltonian.

Next, we use Adiabatic Quantum Computation (AQC) to find the ground state of the problem Hamiltonian. After repeated measurements, this will provide us with a state that minimizes the number of broken clauses. We can then check if we have arrived at our factors  $p$  and  $q$  by simply multiplying them. If the multiplication does not result in a value equal to  $N$ , we must proceed with our next guess.

We repeat these steps until we arrive at our required factors. We would have at most  $\lfloor (b_N + 1)/2 \rfloor$  guesses, which would not make this method inefficient.

In the following sections, we will examine each step outlined in greater detail.

## Classical Preprocessing

Classical preprocessing refers to the steps required to construct as well as minimize the computational complexity of the clauses required.

## Cost Function

In order to convert the semiprime factorization problem to an optimization problem, we must construct a function over our binary variables

$$C : \{0, 1\}^m \rightarrow \mathbb{R}$$

such that the combination of inputs encoding our solution would have the lowest cost assigned.

In the first AQC implementation for factorization, where the number 21 was factored [3], a very straightforward way to construct such a function was given

$$C = [N - P \times Q]^2$$

where,

$$P = \sum_{k=0}^{b_P-1} 2^k p_k$$

$$Q = \sum_{l=0}^{b_Q-1} 2^l q_l$$

are the binary expansions of  $P$  and  $Q$  using binary variables. It's easy to see that for the correct factors, the cost would be zero. This method however doesn't scale well as the spectrum-width scales as  $N^2$  (cost corresponding with  $p = q = 0$ ).

Another method to construct the cost function used long hand multiplication tables [4]. In this case we build a multiplication table like the one for 143 in Figure 2. Then, for each column we can define a cost function. For column  $b_2$ , the variables must satisfy

$$p_2 + p_1 q_1 + q_2 + z_{12} = 1 + 2z_{23} + 4z_{24}$$

which would give us the cost function

$$C_2 = (p_2 + p_1 q_1 + q_2 + z_{12} - 1 - 2z_{23} - 4z_{24})^2$$

The complete cost function would be the sum of the costs associated with each column.

$$C = \sum_k C_k$$

	$b_7$	$b_6$	$b_5$	$b_4$	$b_3$	$b_2$	$b_1$	$b_0$
Multiplier					1	$p_2$	$p_1$	1
					1	$q_2$	$q_1$	1
Binary-multiplication					1	$p_2$	$p_1$	1
				$q_1$	$p_2 q_1$	$p_1 q_1$	$q_1$	
			$q_2$	$p_2 q_2$	$p_1 q_2$	$q_2$		
		1	$p_2$	$p_1$	1			
Carry	$z_{67}$	$z_{56}$	$z_{45}$	$z_{34}$	$z_{23}$	$z_{12}$		
	$z_{57}$	$z_{46}$	$z_{35}$	$z_{24}$				
Product	1	0	0	0	1	1	1	1

Figure 2: Multiplication table for 143 using binary variables for factors as well as carries.

## Rule based variable reduction

This method not only controls the spectrum-width but also enables us to use classical preprocessing through logical deductions to reduce the number of variables needed to factor a given number [5]. This reduction could even make the number of variables less than the size of the problem  $b_N$ . To achieve this, we need to find a set of rule-templates that can be tested on our set of equations in polynomial time and apply suitable substitutions. I have identified an (incomplete) set of rules, given in Table 1

Using a python script, a sample output for a small number is given below

Rule	Variables Reduced
$\sum_{i=0}^n a_i A_i = 0 \implies A_i = 0$	$n$
$A = 1$	1
$A + B = 1 \implies A = \neg B$	1
$(aA = bB) \wedge (a = b) \implies A = B$	1
$A + B = 2X \implies A = B = X$	2
$A + B + 1 = 2X \implies (A = \neg B) \wedge (X = 1)$	2
$A + B + 2C = 2X + 1 \implies (A = \neg B) \wedge (C = X)$	2
Minmax Mismatch*	1

\* For an equation written in the form  $\sum_i a_i A_i + c = \sum_i b_i B_i + d$ , let  $b_j = \max(\{b_i\})$ , then if  $\max(\sum_i a_i A_i + c) = \sum_i a_i + c < b_j + d \implies B_j = 0$ . This rule doesn't completely eliminate the equation.

Table 1: A small set of rules for equation patterns encountered frequently



## N=143, before Optimization

```
..--==Opt100.00%==--.. <N = 143 [1, 1, 1, 1, 0, 0, 0, 1] >
>>> P array : [p0, p1, p2, p3]
>>> Q array : [q0, q1, q2, q3]
>>> Equations :
Removed : []
0000 : 1*p0*q0=+1
0001 : 1*p0*q1+1*p1*q0=2*z_1_2+1
0002 : 1*p0*q2+1*p1*q1+1*p2*q0+1*z_1_2=2*z_2_3+4*z_2_4+1
0003 : 1*p0*q3+1*p1*q2+1*p2*q1+1*p3*q0+1*z_2_3
      =2*z_3_4+4*z_3_5+8*z_3_6+1
0004 : 1*p1*q3+1*p2*q2+1*p3*q1+1*z_2_4+1*z_3_4
      =2*z_4_5+4*z_4_6+8*z_4_7+0
0005 : 1*p2*q3+1*p3*q2+1*z_3_5+1*z_4_5=2*z_5_6+4*z_5_7+0
0006 : 1*p3*q3+1*z_3_6+1*z_4_6+1*z_5_6=2*z_6_7+0
0007 : 1*z_4_7+1*z_5_7+1*z_6_7=+1
```

## N=143, after Optimization

```
..--==Opt10.00%==--.. <N = 143 [1, 1, 1, 1, 0, 0, 0, 1] >
>>> P array : [1, p1, p2, 1]
>>> Q array : [1, 1-p1, 1-p2, 1]
>>> Equations :
Removed : [0, 1, 2, 7, 6, 5, 4]
0003 : 1*p2+1*p1=2*p2*p1+1
```

These rules are able to eliminate most of the variables, reducing the complexity of the problem considerably.

Semiprimes generated by the multiplication of the first 1000 primes were tested. Figure 3 shows us the scaling law of the same.

## Limit testing of the rule based approach

As there is no generalization of the rules nor is there a complete set of rules provided somewhere, various efforts to reduce the number of variables into the sub- $b_N$  regime have been tried in hopes of reducing the complexity to make it possible to factor a large semiprime with the hardware and technology

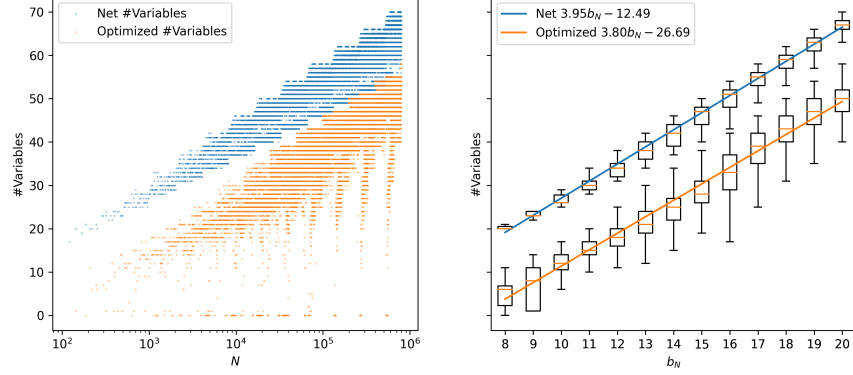


Figure 3: Scaling of the number of variables with the size of the problem  $b_N$

available presently. For example, in a 2021 article [6], Figure 4 was reported.

For example take the final equation that was left after optimization of 143.

$$A + B = 2AB + 1$$

we could in this case set  $B = \neg A$  and completely remove all variables from the problem, in which case the classical preprocessing was able to give us the factors of the number without even needing to go to the AQC part.

But unable to generalize the equations, I took a different approach to test the limits to which this preprocessing idea could be taken. To see how this approach works, lets take a simple rule from the table

$$A + B = 2X$$

or the cost function

$$C = (A + B - 2X)^2$$

Lets make a table with all the possible inputs and their associated cost.

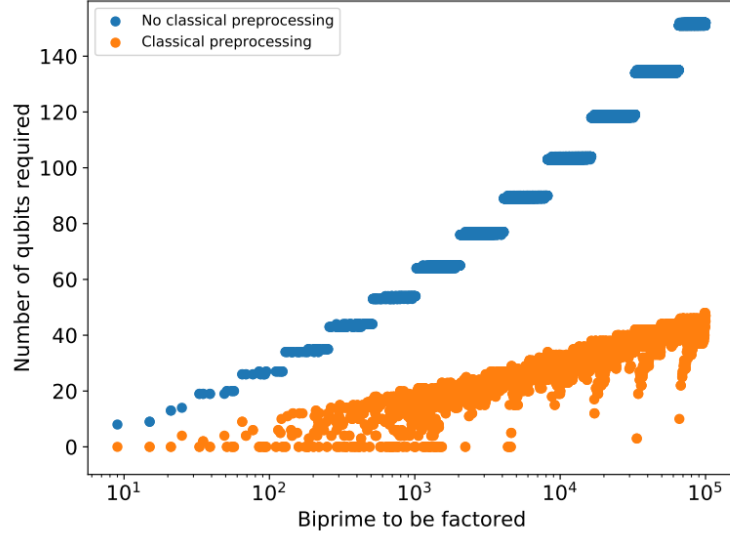


Figure 4: Scaling law using a different set of rules

A	B	X	Cost
0	0	0	0
0	0	1	4
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	4
1	1	1	0

It can be seen from the table that the zero cost corresponds with equating the variables with each other

$$A = B = X$$

which results in the reduction of 2 variables.

A similar approach is taken for any equation encountered. For example take the equation

$$A + B = 2C + 1$$

and its associated cost function

$$C(A, B, C) = (A + B - 2C - 1)^2$$

We can then create a table with all the combinations of inputs and costs, which in this case would be

A	B	C	Cost
0	0	0	1
0	0	1	9
0	1	0	0
0	1	1	4
1	0	0	0
1	0	1	4
1	1	0	1
1	1	1	1

We can then filter out all the combinations that do not satisfy the clause, which leaves us with only two combinations.

A	B	C
0	1	0
1	0	0

Noticing that  $C$  is always zero, we can completely remove this variable from all the equations. Also as  $A + B = 1$ , we can set  $B = \neg A$  and replace  $B$  everywhere as well. The value of  $B$  can be recovered after the completion of AQC, its value being linked to the value of the variable  $A$ . Thus, we have reduced the number of variables required by 2.

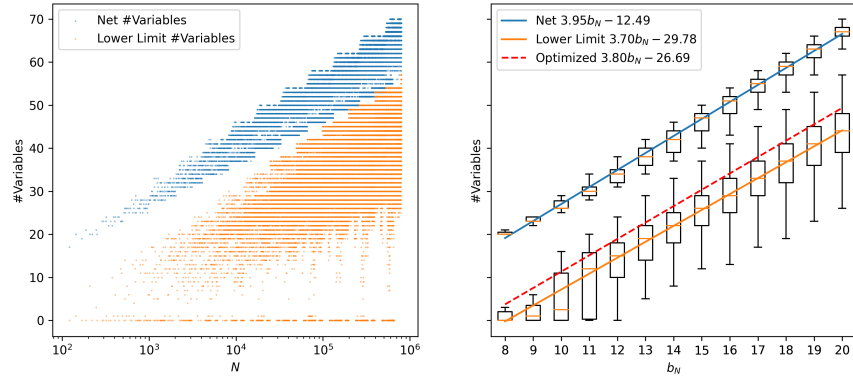
When  $B$  is replaced, it is replaced wherever it occurs in any of the clauses for a semiprime. Doing so starts a domino reaction with the substitution causing changes to other equations that could then be retried with the possibility of more variables being reduced.

It is not necessary for all the variables involved to be reduced in such a way, for example in the case of  $A + 2B + 4C = D + E$ , in which case the only reduction that can be made would be to set  $C$  to zero.

We have not considered substitutions of variables for testing as that would completely reduce all the variables.

This method is not efficient, and in fact itself a satisfiability problem. This method wouldn't be a part of our rule based variable reduction, and has only been introduced to check the limits to which the classical processing could reduce the problem.

Using a Rust program, the generated semiprimes were tested again.



It can be seen from the boxplot that the simple rules introduced before was able to take us very close to the limit. But it also shows that this method cannot be used to take make the variables less than  $b_N$ . It is technically possible to remove any of the carries left in the end as they always occur in pairs, but doing so would increase the spectrum-width that we were trying to control in the first place.

Any similar approach that reduces the spectrum-width would also face the same downfall. For example, we could have tried to take non-binary variables for the carries that would only occur between columns that are next to each other in the multiplication table (carry aggregation) [7], but

that seems to be an even looser condition as all it cares about are the upper and lower limits.

## Adiabatic Quantum Computation

### Mathematics of AQC

AQC works due to the Adiabatic Theorem that states that a system evolving due to a time varying hamiltonian  $\hat{H}(t)$ , which doesn't have a degenerate ground state (except possibly as  $t \rightarrow \tau$ ), would stay in the instantaneous groundstate  $|0(t)\rangle$  in the absence of external disturbances if it starts in the groundstate  $|0(0)\rangle$ , provided that the hamiltonian changes *sufficiently* slowly. [8, 9]

Let the hamiltonian vary over a time  $\tau$ , then in terms of the dimensionless parameter  $s = t/\tau$ , the asymptotic form of the statevector in the case of large  $\tau$  is given by

$$\begin{aligned} |\hat{\psi}(s)\rangle &= \sum_j c_j(s) e^{-i\tau\phi_j(s)} |\hat{j}(s)\rangle \\ c_0(s) &\approx 1 + \mathcal{O}(\tau^{-2}) \\ c_{j \neq 0}(s) &\approx \frac{i}{\tau} [A_j(0) - e^{i\tau[\phi_j(s) - \phi_0(s)]} A_j(s)] + \mathcal{O}(\tau^{-2}) \end{aligned}$$

where  $\phi_j(s) = \int_0^s ds' \epsilon_j(s')$ ,  $\Delta_j(s) = \epsilon_j(s) - \epsilon_0(s)$  and

$$A_j(s) = \frac{1}{\Delta_j(s)^2} \left\langle j(s) \left| \frac{d\hat{H}(s)}{ds} \right| 0(s) \right\rangle$$

The condition for adiabatic evolution is given by the smallness of excitation probability

$$\begin{aligned} \tau &\gg |A_j(s)| \\ \frac{1}{\Delta_j(t)^2} \left| \left\langle j(t) \left| \frac{d\hat{H}(t)}{dt} \right| 0(t) \right\rangle \right| &= \delta \ll 1 \end{aligned}$$

$$\begin{aligned}
\delta &= \frac{1}{\Delta_1(t)^2} \left| \left\langle 1(t) \left| \frac{d\hat{H}(t)}{dt} \right| 0(t) \right\rangle \right| \\
&\leq \frac{E}{\tau g_{\min}^2} \\
\implies \tau &\gg \frac{E}{g_{\min}^2}
\end{aligned}$$

where  $E = \max_s \left\langle 1(s) \left| \frac{d\hat{H}(s)}{dt} \right| 0(s) \right\rangle$  and  $g_{\min} = \min_s \epsilon_1(s) - \epsilon_0(s)$

## Setup

For a satisfiability problem with  $n$  variables  $x_i$ , a system with  $n$  qubits is initialized to the ground state of the initial hamiltonian

$$\hat{H}_0 = g \sum_i \hat{\sigma}_x^{(i)}$$

where  $\hat{X}^{(i)}$  is an operator  $\hat{X}$  acting on the  $i^{\text{th}}$  qubit and  $g$  is a constant controlling the initial strength of the hamiltonian.

System is initialized in the ground state

$$\begin{aligned}
|\psi_0\rangle &= \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]^{\otimes n} \\
&= \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n} (-1)^{b(j)} |j\rangle
\end{aligned}$$

where  $b(j)$  is the parity of  $j$ .

## Observables

Any binary variable  $x_i$  that appears in the above clauses can be mapped to the  $i^{\text{th}}$  qubit using the observable

$$\begin{aligned}
\hat{X}_i &= \frac{1 - \hat{\sigma}_z^{(i)}}{2} \\
&= 0 |0\rangle \langle 0| + 1 |1\rangle \langle 1|
\end{aligned}$$

such that the measurement gives us the value of the binary variable at the end.

## Problem Hamiltonian

For each clause  $C_i$ , the associated cost hamiltonian can be constructed by simply substituting the observable at every occurrence of the variable and multiplying any constants with identity. For example the clause

$$C = (A + B - 2X - 1)^2$$

would be converted to

$$\hat{H}_C = \left[ \hat{A} + \hat{B} - 2\hat{X} - \hat{I} \right]^2$$

such that

$$\hat{H}_C |x_i\rangle = C(x_i) |x_i\rangle$$

Every clause is converted to a hamiltonian in a similar way and our problem hamiltonian is the sum of the individual cost hamiltonians.

$$\hat{H}_p = \sum_i \hat{H}_{C_i}$$

## Annealing Schedule

The system would evolve according to the Time-Dependent Schrodinger Equation

$$\frac{d}{dt} |\psi(t)\rangle = -i\hat{H}(t) |\psi(t)\rangle$$

where we have set  $\hbar$  to 1. The time dependent hamiltonian is given by

$$\hat{H}(t) = f_\tau(t)\hat{H}_0 + g_\tau(t)\hat{H}_p$$

where  $f_\tau$  and  $g_\tau$  are non-increasing and non-decreasing functions denoting the strengths of the initial and problem hamiltonian respectively with the time scale parameter  $\tau$ . For simplicity, I'll go with linear interpolation of these hamiltonians i.e

$$\hat{H}(t) = \left(1 - \frac{t}{\tau}\right) \hat{H}_0 + \frac{t}{\tau} \hat{H}_p$$



## Choice of spectrum-width

From the equation

$$\tau \gg \xi = \frac{E}{g_{\min}^2}$$

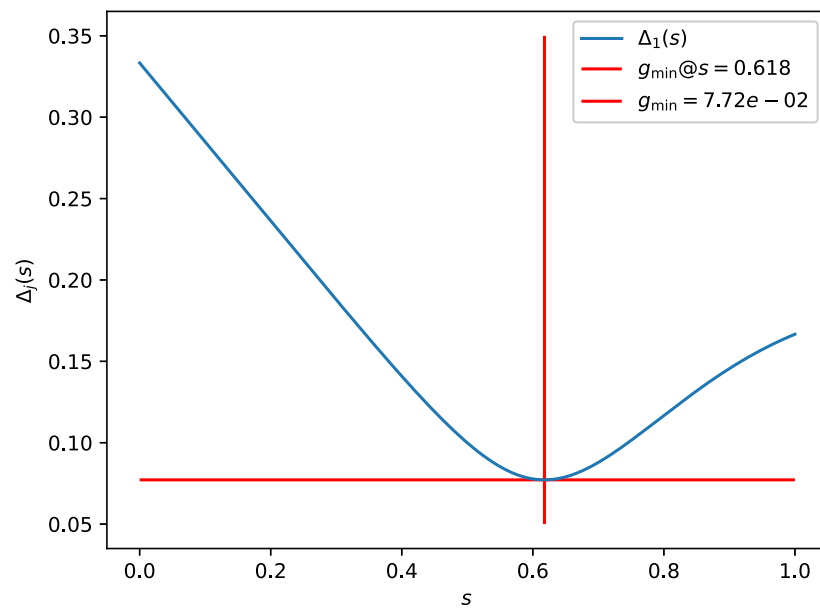
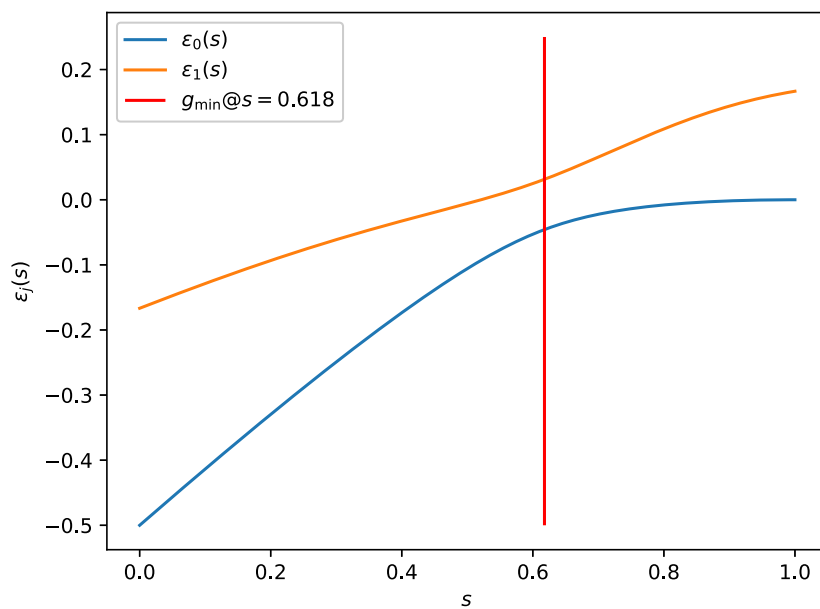
we can see that if we scale the spectrum by a factor of  $k$

$$\begin{aligned}\xi' &= \frac{E'}{g_{\min}'^2} \\ &= \frac{kE}{k^2 g_{\min}^2} \\ &= \frac{\xi}{k}\end{aligned}$$

leads to reduction of time required by a factor  $k$ . So, we'll restrict the spectrum-width to 1 and study the scaling relations.

## Level Diagrams

The figure below shows a level diagram of the first two eigenvalues of  $\hat{H}(t)$  constructed for  $N = 247$  factorization of which requires only 3 qubits, the location of  $g_{\min}$  is shown. The spectrum width  $\epsilon_{\max} - \epsilon_{\min}$  has been restricted to 1.



In the above diagram, the energy difference between the 1<sup>st</sup> excited state and the ground state is plotted. The smaller the energy difference the slower the strength function must change in the optimum schedule.

## Simulation

The evolution of our statevector can be numerically simulated using various methods such as the Crank-Nicolson method which is numerically stable and gives a unitary evolution which preserves the norm of our wavefunction.

For a timestep  $\delta t$  the Crank-Nicolson scheme can be written as

$$|\psi(t + \delta t)\rangle = \left( \hat{I} + \frac{i\hat{H}(t + \delta t)\delta t}{2} \right)^{-1} \left( \hat{I} - \frac{i\hat{H}(t)\delta t}{2} \right) |\psi(t)\rangle$$

But I found that my implementation of the same in Julia was very slow and a finite-difference implementation of the same using a fraction of the time-step yielded better results.

$$|\psi(t + \delta t)\rangle = \left( \hat{I} - i\hat{H}(t)\delta t \right) |\psi(t)\rangle$$

The wavefunction was re-normalized every few hundred time steps.

## Observations

AQC for 30 semiprimes, ranging from 247 to 579349, was simulated. The minimum time  $\tau$  required for the system to have a 95% probability of the correct answers being measured was calculated.

The simulations showed a nice inverse relation between  $g_{\min}$  and  $\tau$  as shown in Figure 5.

The scaling of  $\tau$  with the number of qubits was also plotted in Figure 6, but due to the lack of data points corresponding to  $\geq 8$  qubits, commenting about how the growth of time as a function of the qubits involved was not possible.

The points have different colours representing the different number of clauses involved in the problem hamiltonian, but even this classification didn't seem to give any meaningful observation.

For the case of linear interpolation between initial and problem hamiltonians, an exponential scaling is to be expected [9]. Ideally we would be able to derive the scaling law between  $\tau$  and  $b_N$  using this.

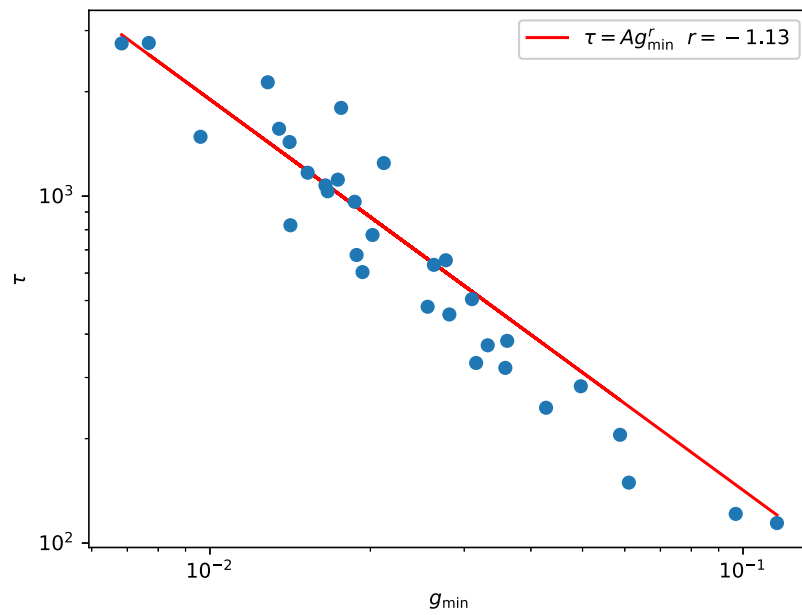


Figure 5: Inverse relation due to restriction of spectrum-width to 1

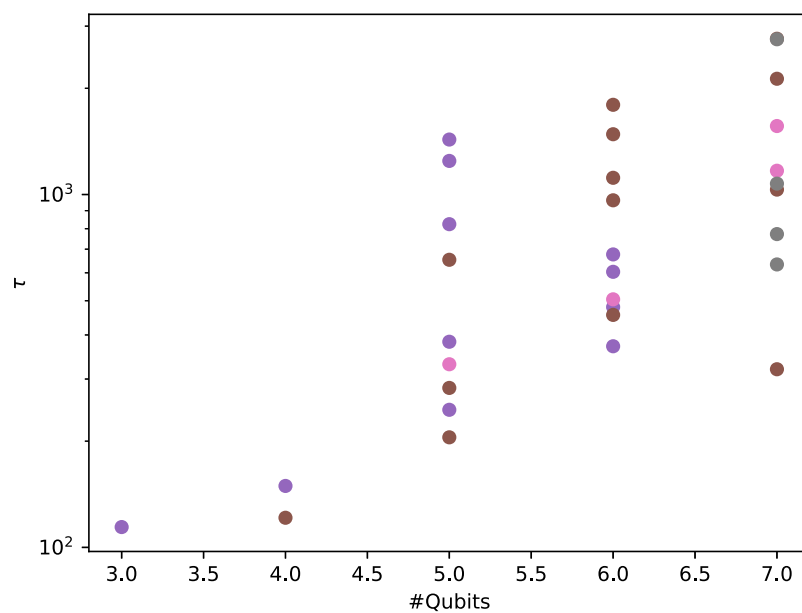


Figure 6: Time to anneal for 30 different semiprimes simulated

## Future Work

### Choice of Annealing Schedule

It was possible to have a variety of different annealing schedules  $f_\tau(t)$  and  $g_\tau(t)$  with the only restricting being that they be non-increasing and non-decreasing respectively. Furthermore, as our problem hamiltonian is made up of many hamiltonians formed from individual clauses, it is possible to associate a strength function with every hamiltonian which would increase independently from each other.

### Derivation of the perfect schedule

If we are able to derive a lower bound for the 1<sup>st</sup> excitation energy,  $g_{b_N}(s)$ , such that for any semiprime of length  $b_N$  would be bounded from below as

$$\forall s \in [0, 1] (g_{b_N}(s) \leq \Delta_1(s))$$

then we'd be able to use this lower bound that would be the most efficient schedule, which could then be used to calculate a scaling relation between  $b_N$  and the time required to anneal  $\tau$ .

## Conclusion

It can be seen that even after figuring out the perfect set of rules, making the number of qubits required for factorizing a number of length  $b_N$  couldn't be made sub- $b_N$ .

Linear interpolation is not a good schedule for the semiprime factorization problem and may result in exponential time complexity. Better schedules, preferably one derived from finding a theoretic lower bound, would have to be used to verify if we can make a polynomial time complexity factorization algorithm.

## References

- [1] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algo-

- rithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001.
- [2] Ari Mizel, Daniel A Lidar, and Morgan Mitchell. Simple proof of equivalence between adiabatic quantum computation and the circuit model. *Physical review letters*, 99(7):070502, 2007.
  - [3] Xinhua Peng, Zeyang Liao, Nanyang Xu, Gan Qin, Xianyi Zhou, Dieter Suter, and Jiangfeng Du. Quantum adiabatic algorithm for factorization and its experimental implementation. *Physical review letters*, 101(22):220405, 2008.
  - [4] Ralf Schützhold and Gernot Schaller. Adiabatic quantum algorithms as quantum phase transitions: First versus second order. *Physical Review A*, 74(6):060304, 2006.
  - [5] Nanyang Xu, Jing Zhu, Dawei Lu, Xianyi Zhou, Xinhua Peng, and Jiangfeng Du. Quantum factorization of 143 on a dipolar-coupling nuclear magnetic resonance system. *Physical review letters*, 108(13):130501, 2012.
  - [6] Amir H Karamlou, William A Simon, Amara Katabarwa, Travis L Scholten, Borja Peropadre, and Yudong Cao. Analyzing the performance of variational quantum factoring on a superconducting quantum processor. *npj Quantum Information*, 7(1):156, 2021.
  - [7] Soham Pal, Saranyo Moitra, VS Anjusha, Anil Kumar, and TS Mahesh. Hybrid scheme for factorisation: Factoring 551 using a 3-qubit nmr quantum adiabatic processor. *Pramana*, 92:1–8, 2019.
  - [8] Max Born and Vladimir Fock. Beweis des adiabatensatzes. *Zeitschrift für Physik*, 51(3):165–180, 1928.
  - [9] Satoshi Morita and Hidetoshi Nishimori. Mathematical foundation of quantum annealing. *Journal of Mathematical Physics*, 49(12):125210, 2008.