Factorisation using Quantum Approximate Optimisation Algorithm (QAOA)

Group members: Ho Tai Yung (Victor)
Lam Lap Yu (Michael)
Lau Yik Man (Johann)
Yam Yeuk Pok Josiah
Zheng Kai Yang (Jack)
Group mentor: Lo Yuk Ho

Who presents each part? (provisionally)

Josiah p3-8 (Introduction)

Jack p9-17 (Shor algorithm and limitations), p18-23 (schematic diagram)

Victor p24-27 (QAOA analogy and visualisation), p67-71

Michael p28-58 (VQF principle), p9-10 (shors’ in short, w/ Jack)

Johann p59-66 (comparison of VQF/QAOA and Shor)

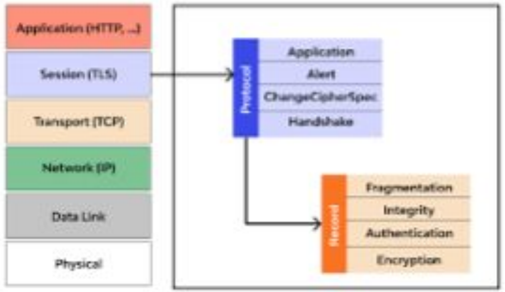
1

Why factorisation?
Why QAOA?

Factorisation and text encryption

RSA algorithm: Examples in cryotography

Secure Internet Communication (HTTPS)



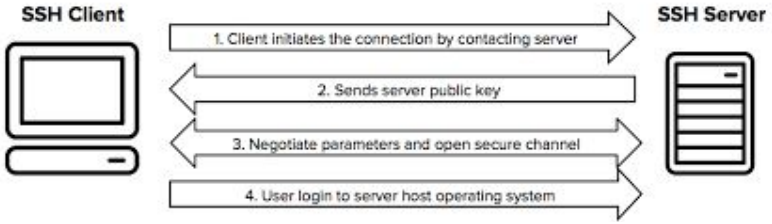
banking system

smart card

VPN

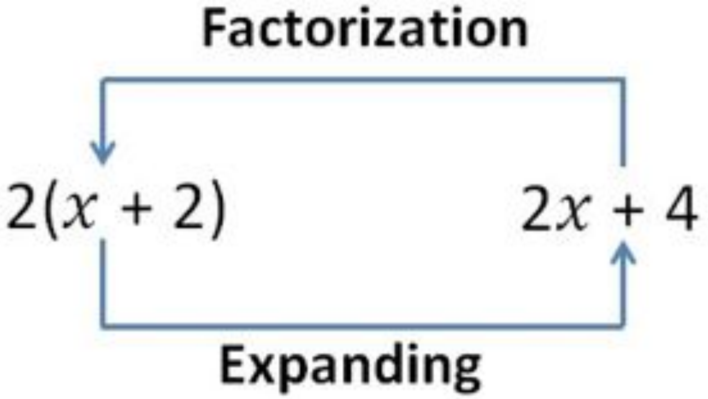
Surfshark®

Secure Shell



Applications of factorisation

For students: just exams?



Just kidding…
there's much more to it

RSA algorithm

- Encryption system based on factoring large semi-prime integers (integers with two prime factors)

71 x 97 = N

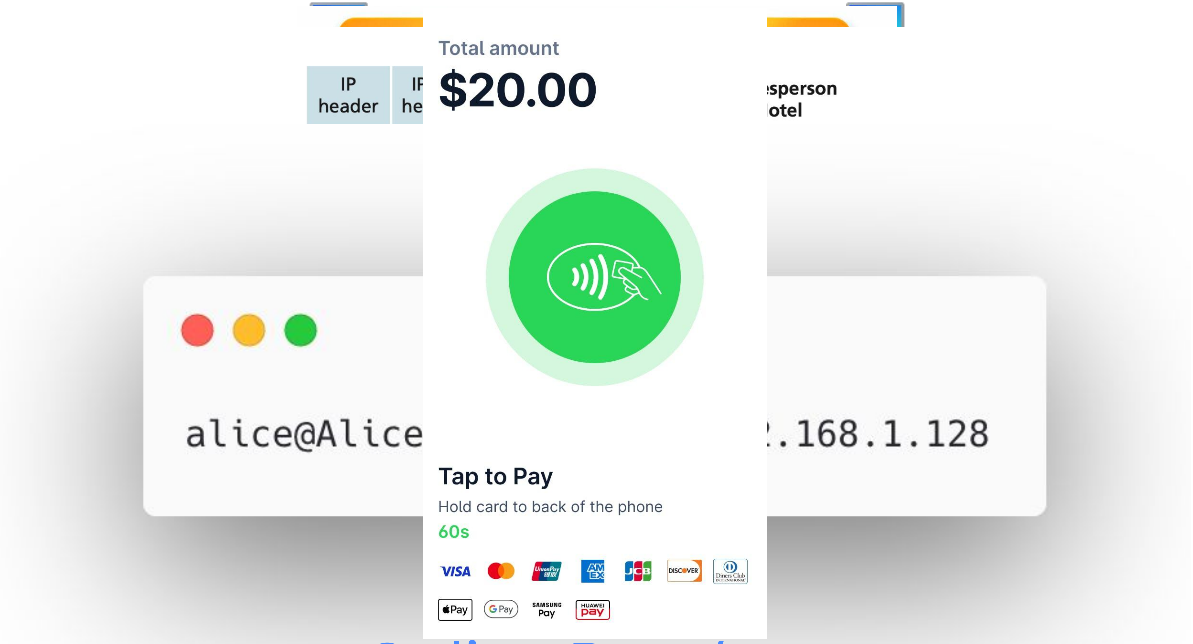
Easy as pi(e)

7493 = P x Q

Ridiculously hard!

Factorisation and text encryption

Examples of factorisation in cryptography



Our data is safe, right?

- Current best classical factoring algorithm for large numbers $>10^{100}$ (general number field sieve) is too inefficient, taking 200k times age of the universe

278970085642197896
764834484013434052
221 = A x B

Get $100 million if you find A and B!

→ How about using quantum algorithms?

2

Shor’s Algorithm

Factorising a large number

- $N = pq$
- p, q are prime numbers
- Find p, q to factorise N

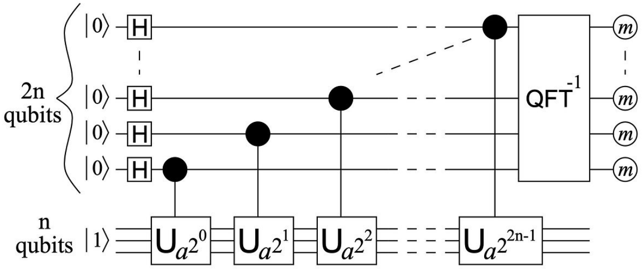
Existing quantum factorisation algorithm: Shor’s algorithm

For some integer a (randomly guessed), $a^x \, mod \, N$ repeats after some integer r iterations. ($a^x mod \, N = a^{x+r} mod \, N$)

For some $a^r \, mod \, N = 1, a^r - 1 = 0 \, mod \, N, \left(a^{r/2} - 1\right)\left(a^{r/2} + 1\right) = 0 \, mod \, N$.
Using quantum phase estimation to find r, if r is even.

→ Greatest common divisor of the two terms with N is highly likely to be a factor of N

Time complexity: $O((\log N)^3)$
CRACKS RSA IN SECONDS!!!



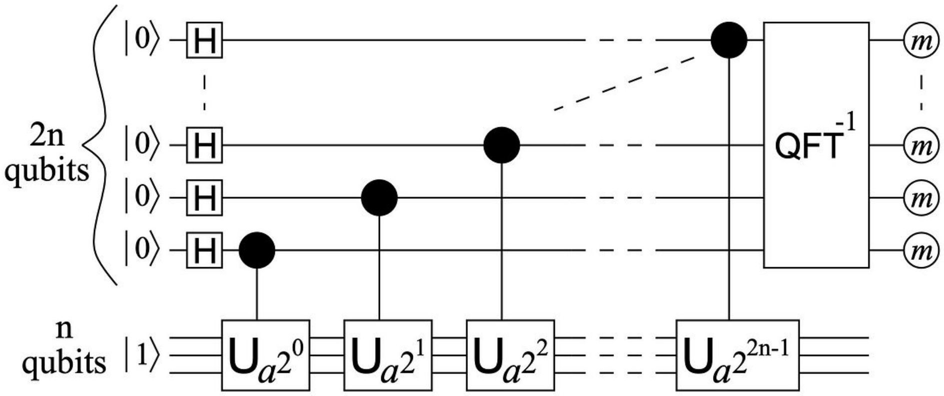
How can we find r???

Quantum phase estimation (QPE)

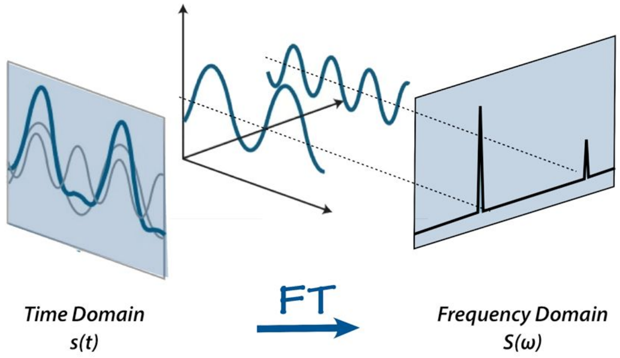
❖ $U\,|y\rangle = |a \cdot y \bmod N\rangle$

→ $|a^1 \bmod N\rangle, |a^2 \bmod N\rangle, |a^3 \bmod N\rangle \ldots |a^{r\text{-}1} \bmod N\rangle$

Use inverse quantum Fourier Transform to extract the period r



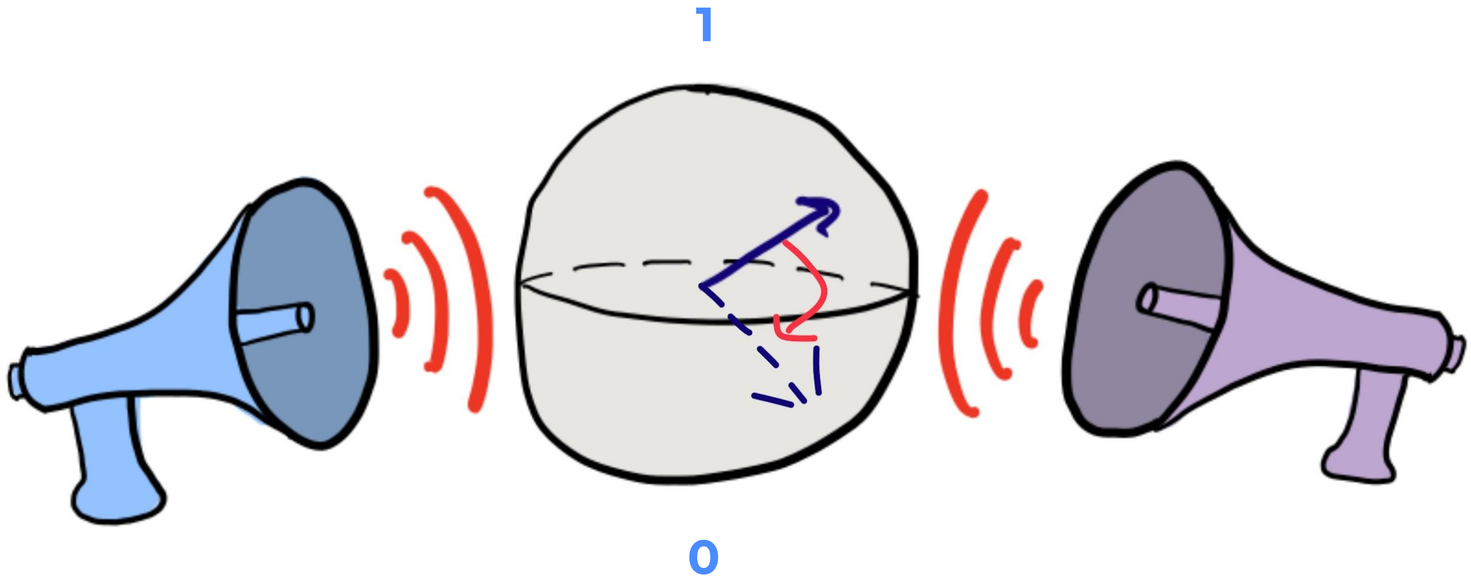
Fourier Transform



Why don’t we just use Shor’s algorithm?

Noisy intermediate-scale quantum (NISQ) software

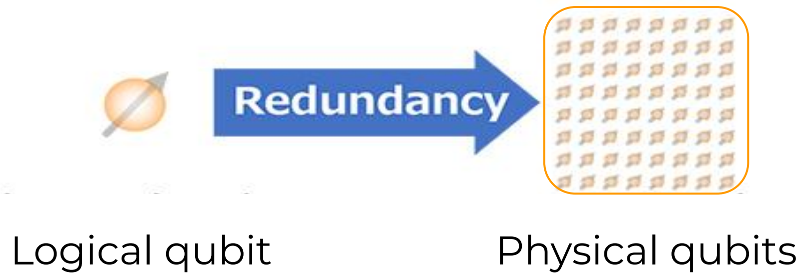
❖ Quantum computer with Error induced by “noise”

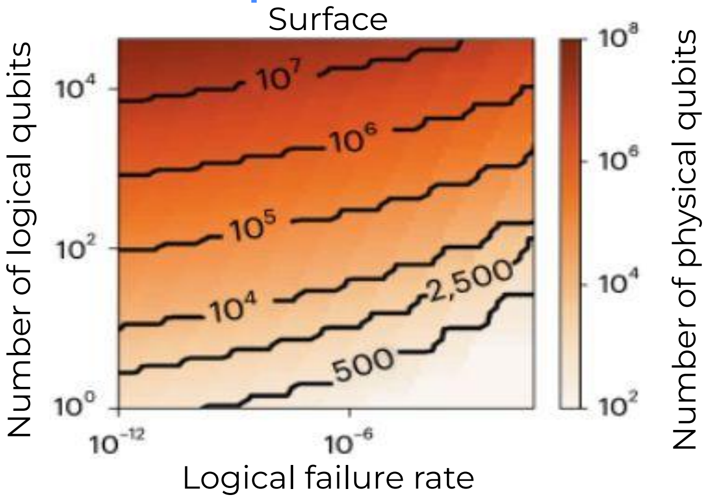


Noisy intermediate-scale quantum (NISQ) software

To solve RSA-2048 by Shor's (factoring digit of 2^2048):
1.5 million physical qubits are needed

Not practical for current quantum computers!



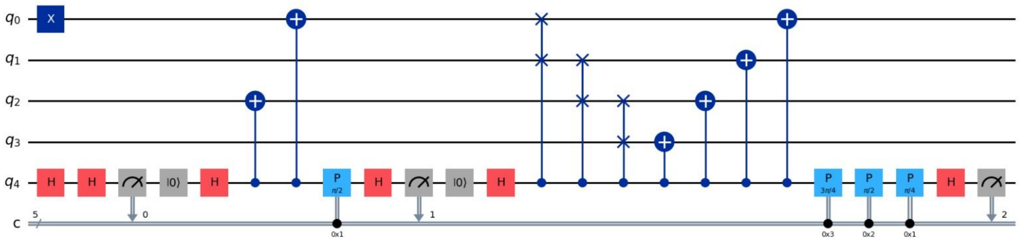


Nat. Phys., **2024**, 20, 1059–1060

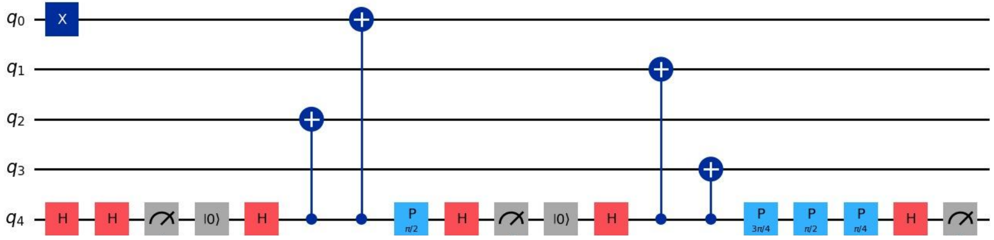
No generalisation of circuits

❖ Modify the circuit for each N

❖ No general solution



◀ N=15



◀ N=21

Factorisation using Quantum Approximate Optimisation Algorithm (QAOA)

Promising & practical algorithm that may beat Shor’s algorithm now? Could it threaten our data security in the near future?

Targets of this project

1. Demonstrate the process of using QAOA via Variational Quantum Factoring (VQF) for factorisation
2. Compare VQF/QAOA to Shor's algorithm in terms of factoring speed (time complexity)
3. Assess the practicality of using QAOA for factorisation applications on current NISQ devices

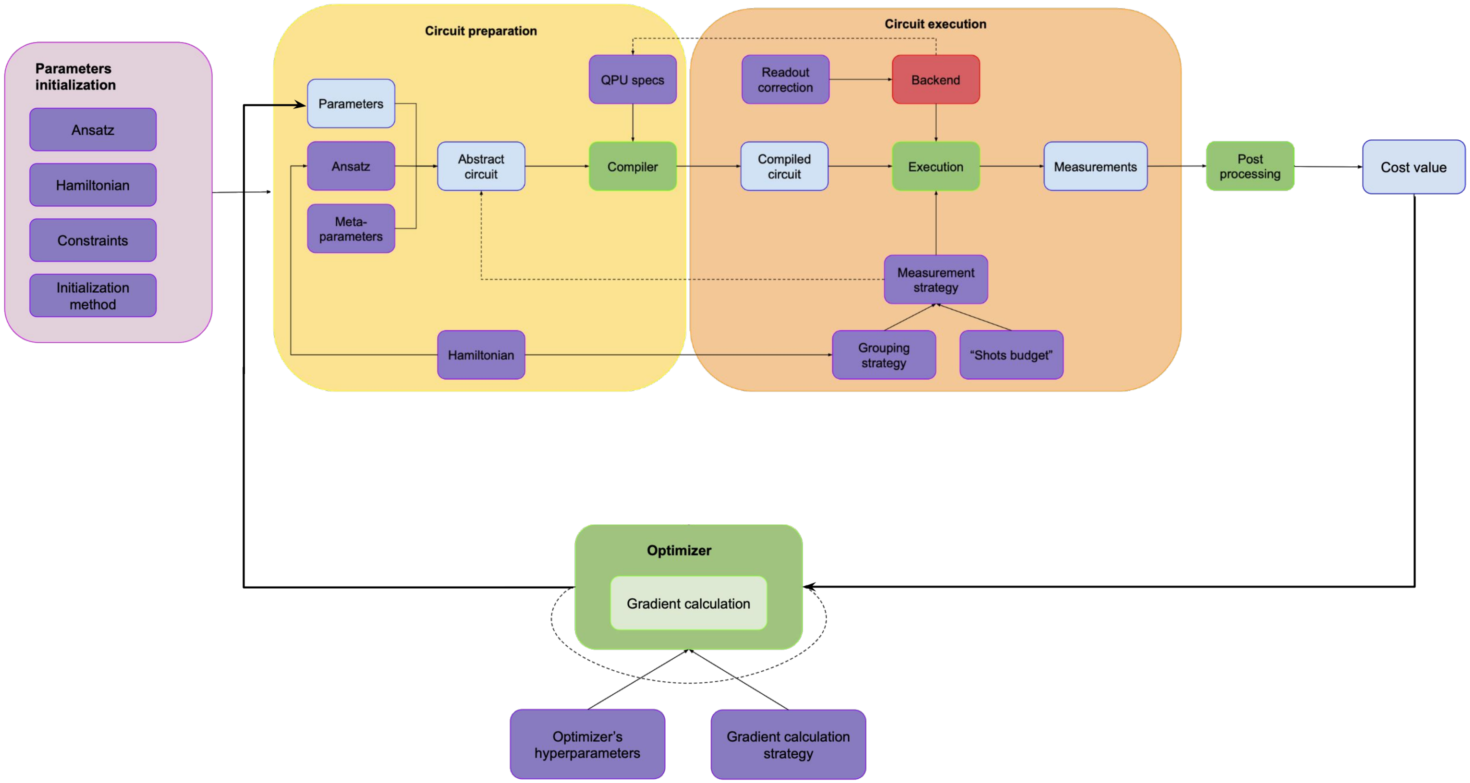
QAOA & noisy quantum hardware

- QAOA: hybrid quantum-classical algorithm to solve some optimisation problems

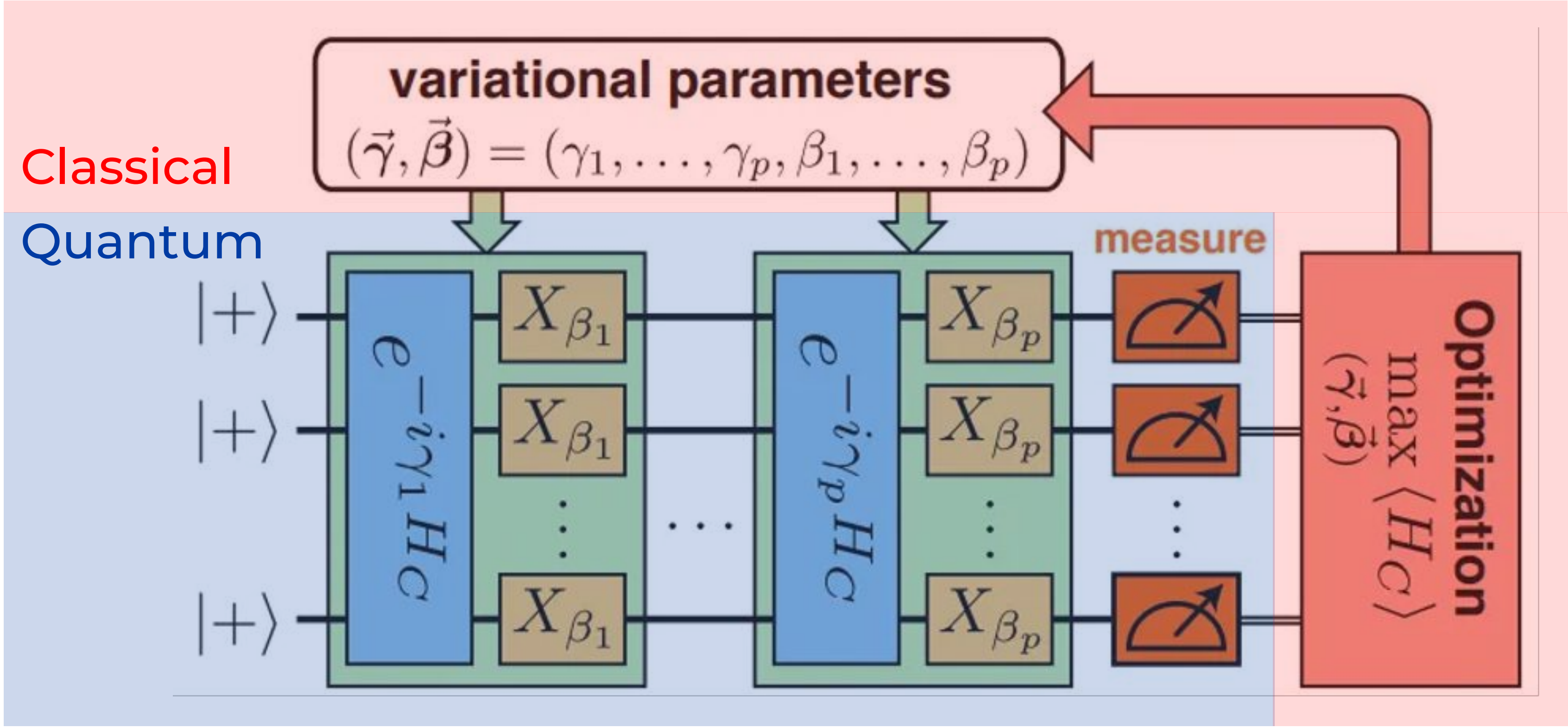
- Unlike Shor’s algorithm, QAOA is designed to function on today’s noisy intermediate-scale quantum (NISQ) hardware

i.e. resilient to some degree of noise; can provide useful outputs even on imperfect hardware

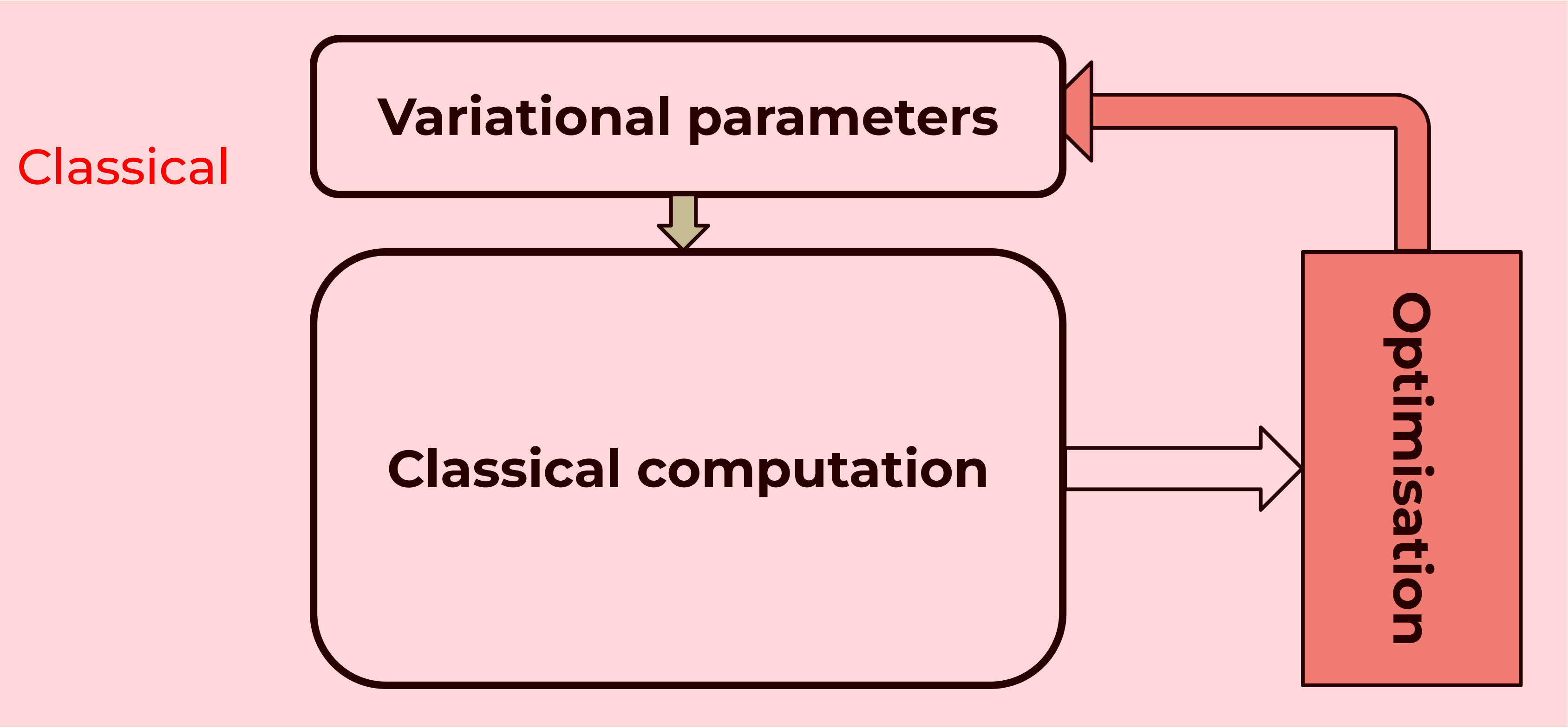
QAOA: full picture



QAOA: overview

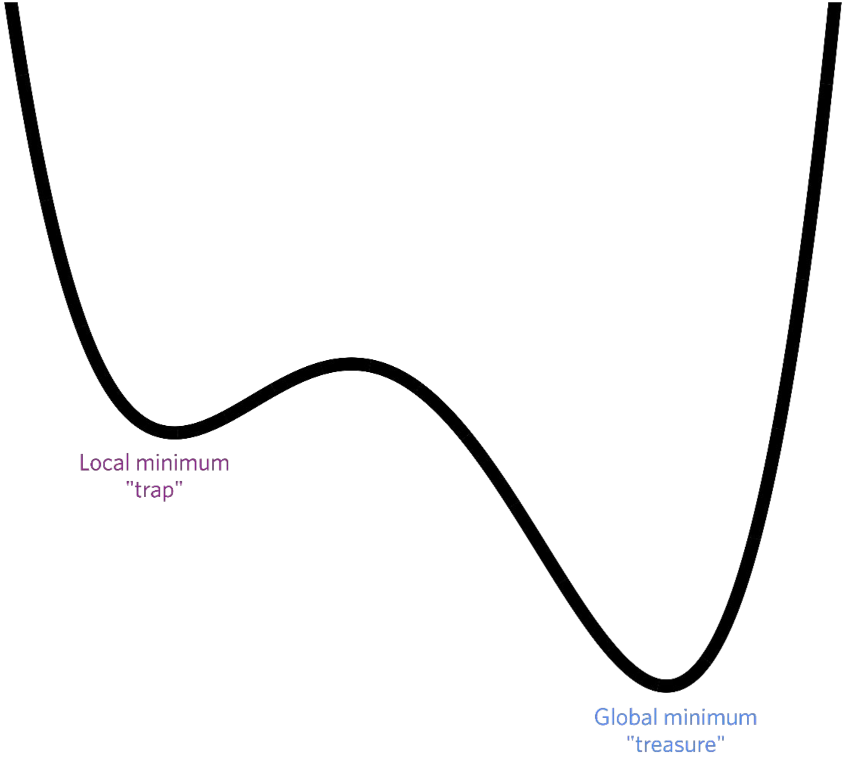


Normal classical optimisation



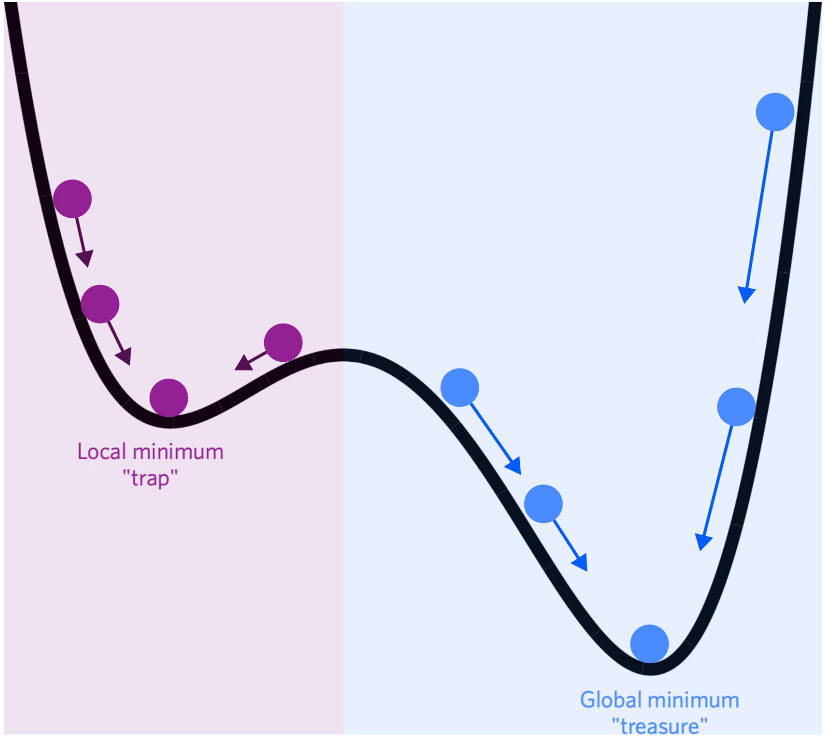
Optimisation: an analogy

1. We need to navigate a landscape and find the absolute minimum point, i.e. the global minimum (treasure)
2. We must avoid other valleys, i.e. local minima (traps)



Optimisation: how does QAOA help?

- Classical algorithms behave like rolling a ball from a random point
- If we start in the blue area, the ball will reach the "treasure" (congratulations!)
- However, if we start in the purple area, we will be stuck in the trap with no way of getting out

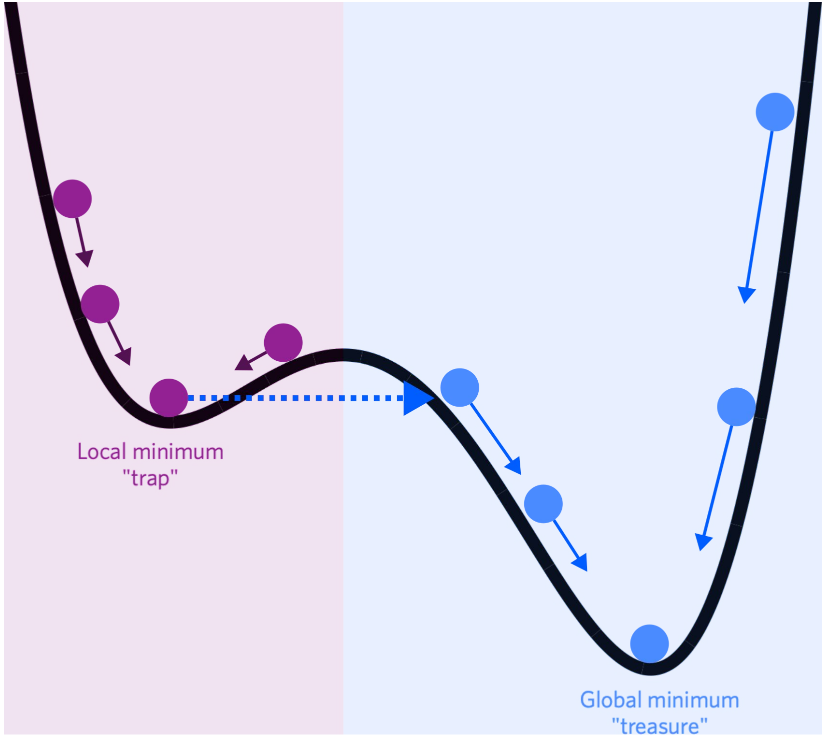


Optimisation: how does QAOA help?

- ❖ This is where the hybrid quantum-classical approach of QAOA helps

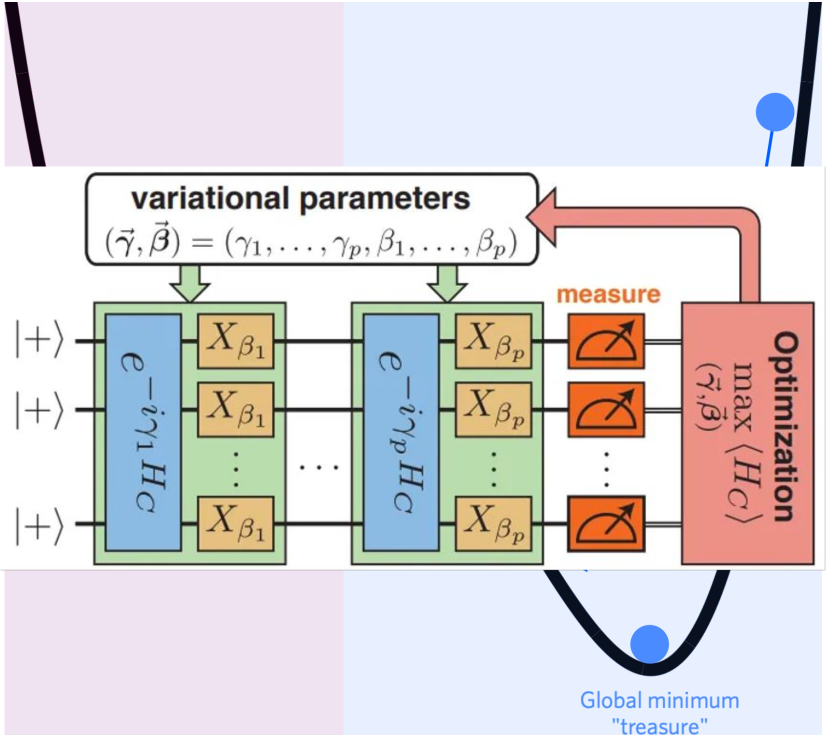
- ❖ Quantum-side noise from NISQ hardware helps classical algorithms starting in the purple area to “tunnel” to the blue area

→ Chance of finding “treasure” increases greatly



Optimisation: more details

- Optimisation in QAOA is based on two variational variables: γ and β
- γ performs a guidance role and leads the ball towards the treasure
- β detects traps and helps with tunneling to avoid them



3

Variational Quantum Factoring (VQF)

A way of applying QAOA

What is VQF?

- Adapts principles from QAOA to solve factoring problems using variational approach
- Suitable for NISQ devices
- Tailored for factoring (whereas QAOA focuses on optimisation problems)
- Differ from QAOA in problem encoding

VQF Part 1:
Factorisation as binary optimisation

To factor $\mathrm{m} = \mathrm{pq}$, we represent it in binary form
(z denotes the carried bits)

$$
\begin{aligned}
m &= \sum_{k=0}^{n_m-1} 2^i m_k, \\
p &= \sum_{k=0}^{n_p-1} 2^i p_k, \\
q &= \sum_{k=0}^{n_q-1} 2^i q_k,
\end{aligned}
$$

1011011 (91)

0111 (7)

1101 (13)

VQF Part 1:
Factorisation as binary optimisation

To factor $m = pq$, we represent it in binary form
(z denotes the carried bits)

Carry out binary multiplication to obtain a system of equations to solve

1	0	1	1	0	1	1
			p_1	p_2	p_3	1
			q_1	q_2	q_3	1
			p_1	p_2	p_3	1
		q_3p_1	q_3p_2	q_3p_3	q_3	0
	q_2p_1	q_2p_2	q_2p_3	q_2	0	0
q_1p_1	q_1p_2	q_1p_3	q_1	0	0	0

$$p_3 + q_3 = 1$$

$$p_2 + q_3p_3 + q_2 = 2z_1$$

$$p_1 + q_3p_2 + q_2p_3 + q_1 + z_1 = 1 + 2z_2 + 2z_3$$

$$q_3p_1 + q_2p_2 + q_1p_3 + z_2 = 1 + 2z_4$$

$$q_2p_1 + q_1p_2 + z_3 + z_4 = 2z_5 + 4z_6$$

$$q_1p_1 + z_5 = 1$$

$$z_6 = 0$$

VQF Part 1:
Factorisation as binary optimisation

When $\mathrm{m} = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
  1011011        (91)
     0111        (7)
x    1101        (13)
 ________
     0111
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
     0111      (7)
x    1101      (13)
---------
     0111
    0000
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
      0111        (7)
x     1101        (13)
 ---------
      0111
     0000
    0111
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
          0111        (7)
x         1101        (13)
_______________
          0111
         0000
        0111
       0111
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
     0111       (7)
x    1101       (13)
 ─────────
     0111
    0000
   0111
  0111
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
__________
      0111
     0000
    0111
+  0111

__________
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
__________
      0111
     0000
    0111
+  0111

__________
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
 ___________
      0111
     0000
    0111
+  0111

 ___________
         1
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
_____________
       0111
      0000
     0111
+   0111

_____________
          1
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
__________
      0111
     0000
    0111
+  0111

__________
        11
```

VQF Part 1:
Factorisation as binary optimisation

When $\mathrm{m} = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
____________
        0111
       0000
      0111
+    0111

____________
          11
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
         0111
        0000
       0111
+     0111
         1       ← Carry forward bits

          011
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
-----------
      0111
     0000
    0111
+  0111
      1     ← Carry forward bits
-----------
       011
```

VQF Part 1:
Factorisation as binary optimisation

When $\mathrm{m} = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
___________
       0111
      0000
     0111
+   0111
      11      ← Carry forward bits
___________
       1011
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
    ____________
           0111
          0000
         0111
+       0111
          11      ← Carry forward bits
    ____________
           1011
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
____________
       0111
      0000
     0111
+   0111
     111     ← Carry forward bits
____________
      11011
```

VQF Part 1:
Factorisation as binary optimisation

When $\mathrm{m} = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
___________
       0111
      0000
     0111
+   0111
     111      ← Carry forward bits
___________
      11011
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
___________
       0111
      0000
     0111
+   0111
    1111      ← Carry forward bits
___________
     011011
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
__________
     0111
    0000
   0111
+ 0111
  1111      ← Carry forward bits
__________
   011011
```

VQF Part 1:
Factorisation as binary optimisation

When $m = 91$, we know that $91 = 7 \times 13$

In binary form, $1011011 = 0111 \times 1101$, perform the multiplication just as we were taught in primary school

```
___________
      0111
     0000
    0111
+  0111
   1111     ← Carry forward bits
___________
   1011011      (We get 91 back)
```

VQF Part 1:
Factorisation as binary optimisation

```
   1011011      (91)        m

      0111      (7)         p
x     1101      (13)        q
----------
      0111
     0000
    0111
+  0111
   1111100                   z (carry bits)
----------
   1011011      (91)        we get m back
```

VQF Part 1:
Factorisation as binary optimisation

We come back to this table…

Now p and q are unknown.

1	0	1	1	0	1	1
			p_1	p_2	p_3	1
			q_1	q_2	q_3	1
			p_1	p_2	p_3	1
		q_3p_1	q_3p_2	q_3p_3	q_3	0
	q_2p_1	q_2p_2	q_2p_3	q_2	0	0
q_1p_1	q_1p_2	q_1p_3	q_1	0	0	0

$$p_3 + q_3 = 1$$

$$p_2 + q_3p_3 + q_2 = 2z_1$$

$$p_1 + q_3p_2 + q_2p_3 + q_1 + z_1 = 1 + 2z_2 + 2z_3$$

$$q_3p_1 + q_2p_2 + q_1p_3 + z_2 = 1 + 2z_4$$

$$q_2p_1 + q_1p_2 + z_3 + z_4 = 2z_5 + 4z_6$$

$$q_1p_1 + z_5 = 1$$

$$z_6 = 0$$

VQF Part 1:
Factorisation as binary optimisation

Generalise to get

$$0 = \sum_{j=0}^{i} q_j p_{i-j} + \sum_{j=0}^{i} z_{j,i} - \sum_{j=1}^{n_c} 2^j z_{i,i+j} - m_i$$

We then associate a clause (constraint) C_i with each equation

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

Then the problem of factoring becomes solving $0 = \sum_{i=0}^{n_c} {C_i}^2$

VQF Part 2:
Simplifying equations

Using classical computers to solve the equations in part 1, for $x, y, x_i \in \{0, 1\}$ and $a, b \in \mathbb{Z}^+$. (a, b are positive integers)

$$xy - 1 = 0 \implies x = y = 1,$$
$$x + y - 1 = 0 \implies xy = 0,$$
$$a - bx = 0 \implies x = 1,$$
$$\sum_i x_i = 0 \implies x_i = 0,$$
$$\sum_{i=1}^{a} x_i - a = 0 \implies x_i = 1.$$

x	y	xy - 1
0	0	-1
0	1	-1
1	0	-1
1	**1**	**0**

In this example, $x=y=1$

VQF Part 2:
Simplifying equations

Number of qubits required is reduced from $O(Nm \log(Nm))$ to $O(Nm)$

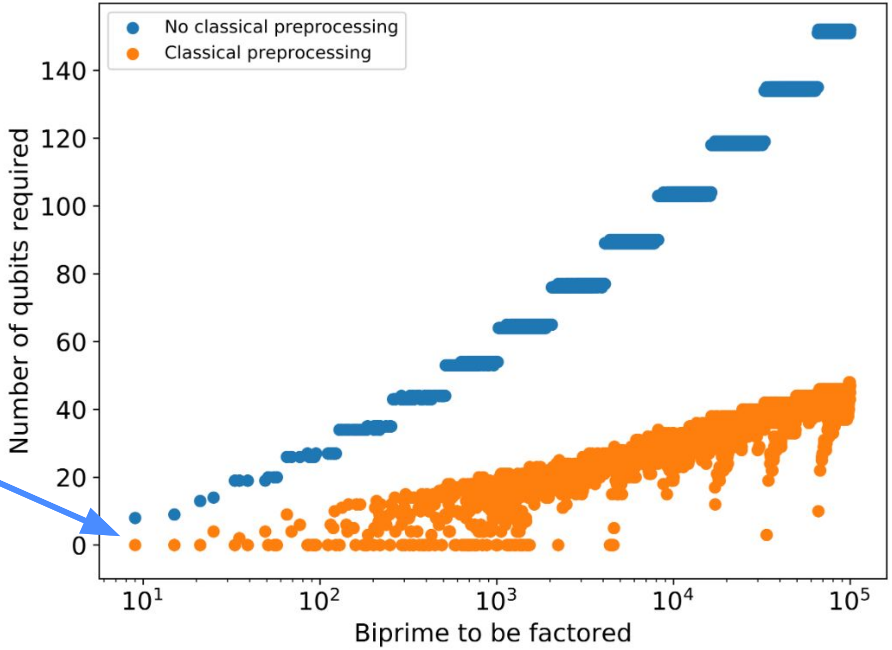
$$xy - 1 = 0 \implies x = y = 1,$$

$$x + y - 1 = 0 \implies xy = 0,$$

$$a - bx = 0 \implies x = 1,$$

$$\sum_i x_i = 0 \implies x_i = 0,$$

$$\sum_{i=1}^{a} x_i - a = 0 \implies x_i = 1.$$



Springer, **2019**, Proceedings 1, 74-85

VQF Part 3:
Constructing an Ising Hamiltonian

We want to solve $0 = \sum_{i=0}^{n_c} C_i^{\,2}$

Let C' be C after applying the classical preprocessing in part 2. $C_i' = 0$

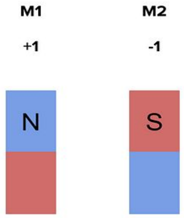
Solutions to $E = \sum_{i=0}^{n_c} C_i'^{\,2}$ correspond to minimisation of classical energy function which has a natural quantum representation $H = \sum_{i=0}^{n_c} \hat{C}_i^2$

VQF Part 3:
Constructing an Ising Hamiltonian

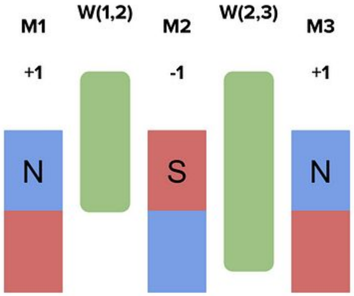
Each Ĉ is obtained by quantising p[i], q[i] and z[j,i], using mapping, where *k* is the bit index.

$$\{p, q, z\} \rightarrow \frac{1}{2}(1 - \sigma^{z}_{\{p,q,z\},k})$$

We have thus encoded the factoring into the ground state of a 4-local Ising Hamiltonian.



$$H = M_1 * M_2$$



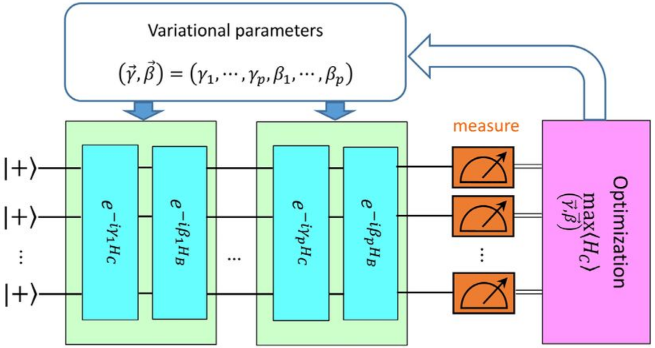
$$H = \sum_{<i,j>} W_{ij} M_i M_j$$

<- what a Hamiltonian is like, each Wij is unknown and we find M such that H is minimised

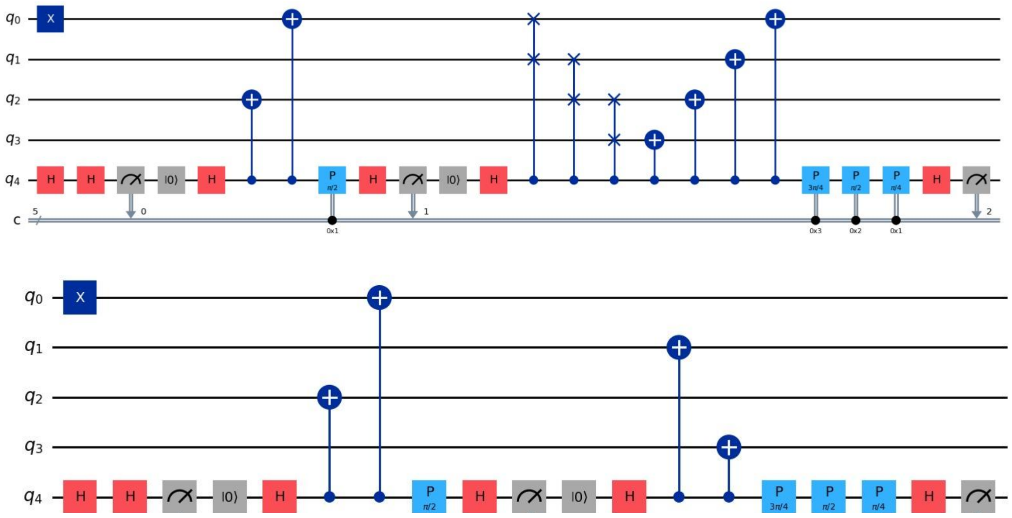
How good is VQF compared to Shor's algorithm?

VQF vs Shor’s algorithm
by circuit construction

VQF

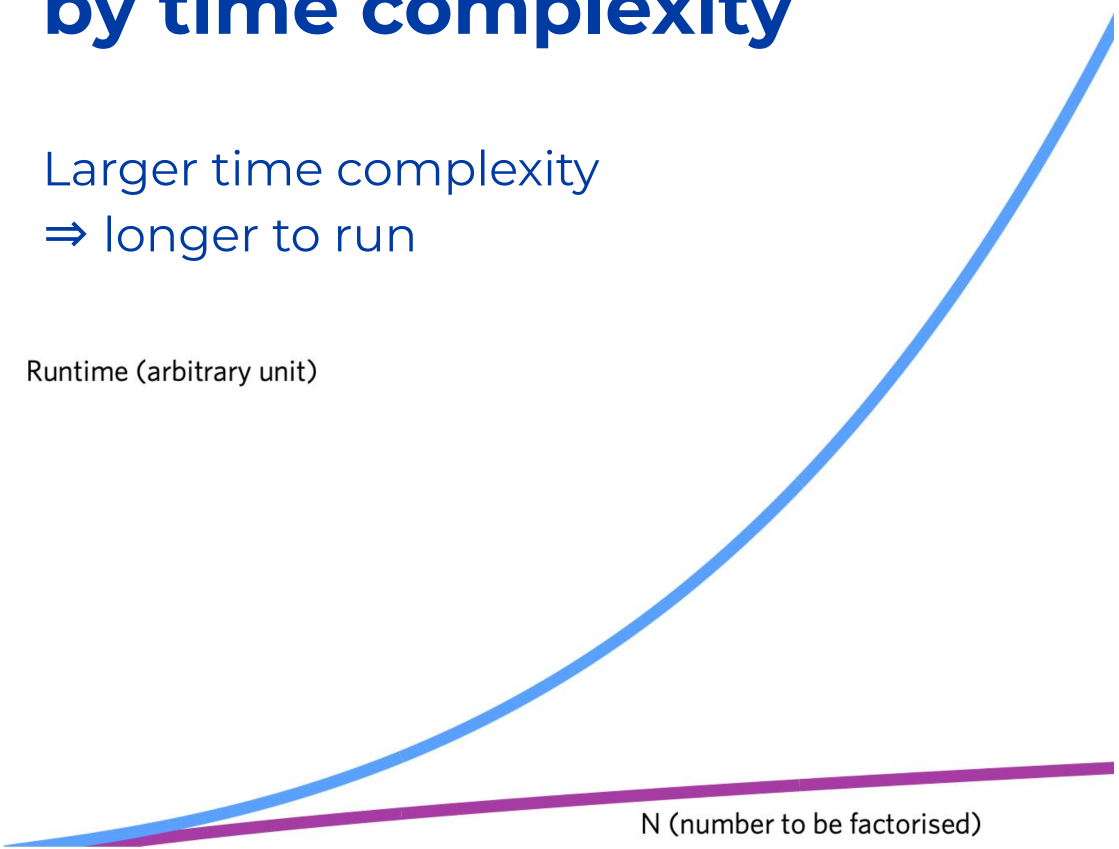


Shor’s algorithm



VQF vs Shor's algorithm
by time complexity

Larger time complexity
⇒ longer to run



VQF

Time complexity unknown
(slower, super-polynomial or sub-exponential)

Shor's algorithm

$O((\log N)^3)$
(faster, sublinear)

VQF vs Shor's algorithm
by practicality for NISQ devices

VQF

- Highly robust against noise
  - It is claimed that noise may even help escape local minima
- Lower qubit requirement

→ More practical for NISQ devices

Shor's algorithm

- Low tolerance against error
- High requirement for qubits

→ Rather impractical for NISQ devices

VQF vs Shor's algorithm
Summary

	VQF	Shor's algorithm
Circuit construction	Can be directly constructed **(Less complex)**	Unique circuit for each N No general method **(More complex)**
Time complexity	super-polynomial / sub-exponential **(Slower)**	$(\log N)^3$ sublinear **(Faster)**
Practicality for NISQ devices	**More practical**	**Less practical**

But how does VQF match up against classical algorithms?

Runtime Comparison

- VQF is less practical than classical algorithms
  - Quadratic sieve (classical) took 2.2 seconds to factorise 2,912,522,065,715,399 on a basic computer
  - VQF struggles to factor fairly small numbers (e.g. 1,207) reliably due to error
- However, its speedup may be possible through development of NISQ devices

Springer, **2019**, Proceedings 1, 74-85

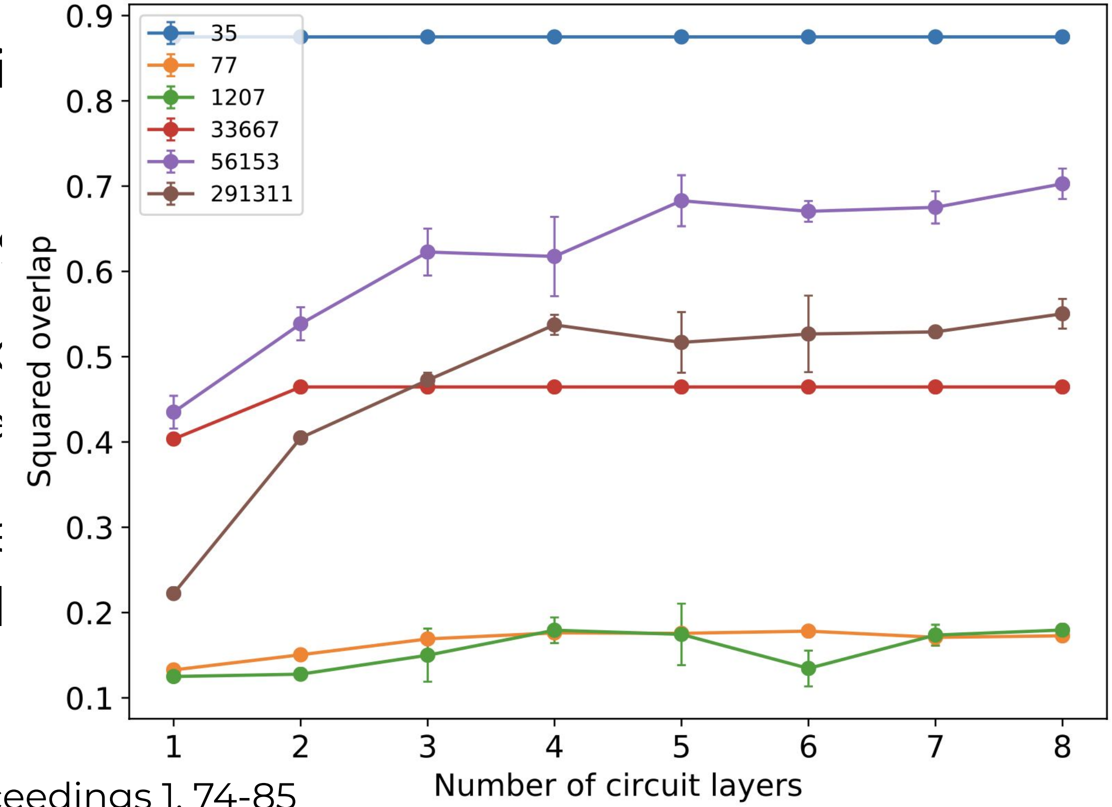
Runtime Comparison

❖ VQF i

➢ Q … torise 2,…

➢ V … 1,207) re…

❖ Howe… devel…



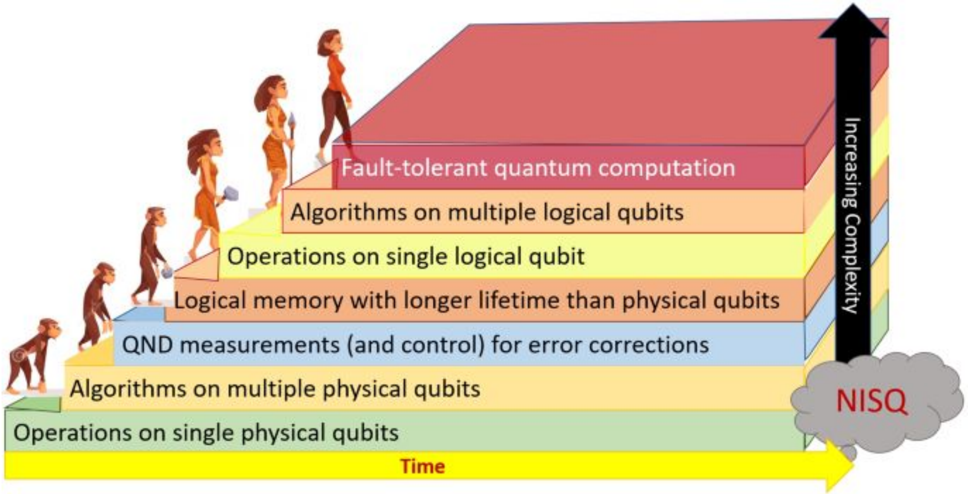
Springer, **2019**, Proceedings 1, 74-85

4

Conclusion

Future outlook on QAOA

- Challenges: NISQ hardware limitations impact the error probability of QAOA circuits and their performance.
- Current quantum systems hold up to ~6000 qubits, which is insufficient to run Shor's algorithm and only provides low performance for QAOA.
- In the future, the number of qubits may skyrocket, meaning QAOA would be more practically used and our data could become less secure.



Efficient factorisation and decryption with QAOA may become possible as NISQ technology advances.

Thank you

References

1. A compact neutral-atom fault-tolerant quantum computer based on new quantum codes. Nat. Phys. 20, 1059–1060 (2024). https://doi.org/10.1038/s41567-024-02480-6

2. E. Anschuetz, J. Olson, A. Aspuru-Guzik, and Y. Cao, Variational Quantum Factoring. Springer. Proceedings 1 , pp. 74–85. (2019)

Q&A

Feel free to ask any questions you may have about our presentation