Crack the Code with Quantum Factorization

K-Quantum Tigers

February 2, 2025

1 Introduction

Shor's algorithm has redefined the landscape of computational complexity, demonstrating that quantum computers can tackle problems far beyond the reach of classical systems. This realization underscores the transformative potential of quantum computing, particularly in the realm of cryptography.

The Quantum Factorization Challenge builds upon this foundation, inviting researchers and practitioners to explore the frontiers of quantum computation. With access to the Quantum Rings Simulator—capable of simulating circuits with hundreds of qubits and executing millions of quantum gate operations—participants are challenged to factor increasingly larger semiprime numbers using pure quantum algorithms.

This challenge is not merely an exercise in computation but a crucial test of quantum feasibility. The ability to factor large semiprimes efficiently threatens the integrity of modern encryption schemes, such as RSA, which rely on the difficulty of this problem for security. By demonstrating the practical capabilities of Shor's algorithm on quantum hardware, we take a step toward understanding both the power and limitations of quantum computation.

In this documentation, we present an in-depth exploration of our approach to implementing and optimizing Shor's algorithm. We discuss the theoretical underpinnings, the architectural considerations, and the novel optimizations applied to improve scalability and accuracy. Through this work, we aim to contribute to the ongoing advancement of quantum algorithms and their implications for future cryptographic paradigms.

2 Algorithm Overview

Shor's algorithm consists of the following steps:

1. Choose a random integer a such that $1 < a < N$.
2. Compute $\gcd(a, N)$. If $\gcd(a, N) \neq 1$, then a factor is found.

3. Use quantum computation to find the period r of the function $f(x) = a^x \mod N$.
4. If r is even and $a^{r/2} \not\equiv -1 \mod N$, then compute $\gcd(a^{r/2} - 1, N)$ and $\gcd(a^{r/2} + 1, N)$, which are factors of N.

3 Code Analysis

The provided implementation of Shor's algorithm follows these steps:

3.1 Quantum Circuit Initialization

We initialize the quantum circuit using Qiskit and QuantumRingsLib.

```
class ShorAlgorithm:
    def __init__(self, config: QuantumConfig = QuantumConfig
        ()):
        self.config = config
        self.provider = QuantumRingsProvider(
            token=config.provider_token,
            name=config.provider_name
        )
        self.backend = self.provider.get_backend(config.
            backend_name)
        self.provider.active_account()
```

This class sets up the quantum provider and backend for executing quantum computations.

3.2 Quantum Fourier Transform (QFT)

QFT is used to determine periodicity in the quantum state.

```
def run_quantum_circuit(self, N: int, a: int) -> Tuple[
    Optional[int], Dict[str, int]]:
    t = math.ceil(math.log(N, 2)) * 2
    n = math.ceil(math.log(N, 2))
    total_qubits = t + n

    q = QuantumRegister(total_qubits, 'q')
    c = ClassicalRegister(t, 'c')
    qc = QuantumCircuit(q, c)

    # Apply Hadamard gates to create superposition
    for i in range(t):
        qc.h(q[i])
    qc.barrier()
```

Here, Hadamard gates create a superposition state, enabling period-finding.

3.3 Inverse Quantum Fourier Transform (IQFT)

The inverse QFT extracts periodicity information:

```python
def iqft(qc: QuantumCircuit, qubits: List[int]) -> None:
    for i in range(len(qubits)):
        qc.h(qubits[i])
        for j in range(i + 1, len(qubits)):
            angle = -2 * math.pi / (2 ** (j - i + 1))
            qc.cu1(angle, qubits[j], qubits[i])
    qc.barrier()
```

IQFT is applied to extract the periodic structure from quantum states.

3.4 Modular Exponentiation

We implement modular exponentiation using controlled unitary operations:

```python
def modular_exponentiation(qc: QuantumCircuit,
    control_register: List[int], target_register: List[int],
    a: int, N: int):
    for j, control in enumerate(control_register):
        factor = pow(a, 2**j, N)
        ModularArithmetic.controlled_modular_multiplication(
            qc, control, target_register, factor, N
        )
    qc.barrier()
```

This ensures that we correctly compute $a^x \mod N$ on the quantum circuit.

4 Lessons and Novel Insights

The implementation of Shor's algorithm provides several key insights:

- **Quantum Fourier Transform (QFT):** The efficiency of QFT is crucial in determining the period accurately. The inverse QFT is used to extract frequency information from the quantum state.
- **Modular Arithmetic in Quantum Circuits:** Optimizing modular exponentiation circuits significantly reduces the number of required gates and enhances performance.
- **Noise and Decoherence Challenges:** In real-world quantum processors, errors due to decoherence must be mitigated using error correction techniques.
- **Potential Cryptographic Impact:** Once large-scale quantum computers are available, RSA and similar cryptographic systems will be vulnerable, necessitating the transition to quantum-resistant cryptographic protocols.

- **QuantumRingsLib Optimization:** The use of ‘QuantumRingsLib‘ ensures that the implementation is optimized for specific quantum hardware.
- **Enhanced Phase Correction in Modular Arithmetic:** Additional phase correction techniques improve computational stability in modular operations.
- **IQFT Precision Improvements:** The implementation uses ‘cu1‘-based phase correction instead of traditional controlled-Rk gates, improving numerical precision.
- **Refined Continued Fraction Processing:** More accurate handling of continued fractions enhances the reliability of period extraction.
- **Adaptive Error Handling and Input Selection:** Mechanisms are included to prevent incorrect period estimation and invalid input selection.

5 Conclusion

Shor’s algorithm exemplifies the power of quantum computing in solving problems that are intractable for classical computers. While current quantum hardware is not yet capable of factoring numbers used in real-world cryptographic applications, rapid advancements in quantum technology suggest that such capabilities may be within reach in the near future. Understanding and refining Shor’s algorithm is critical for both advancing quantum computing and preparing for the post-quantum cryptographic era.