Análise Comparativa do Algoritmo de Shor em Arquiteturas Clássica e Quântica

Caroline D. Gandolfi[1], **Evelyn Henriques Q. V. Costa**[1], **Vitor S. M. Sakai**[1], **Anderson F. P. dos Santos**[1,2], **Raquel C. G. Pinto**[1]

[1]Instituto Militar de Engenharia (IME)
Rio de Janeiro – RJ – Brasil

[2]Venturus Centro de Inovação Tecnológica
Campinas – SP – Brasil

{caroline.gandolfi29,evelynhenriques,vsmsakai,anderson,raquel}@ime.eb.br

Abstract. *Shor's algorithm poses a threat to modern asymmetric cryptography, which relies on the difficulty of factoring large integers. This paper compares its performance in three scenarios: sequential classical simulation, parallel classical simulation with OpenMP, and execution on IBM's quantum hardware. Results show that classical parallelization achieves up to a 7.5x speedup, while current quantum processors remain limited by noise and decoherence. The study highlights the gap between the algorithm's theoretical potential and its practical feasibility, pointing to the technological challenges ahead.*

Resumo. *O algoritmo de Shor ameaça a criptografia assimétrica moderna, baseada na dificuldade de fatoração de grandes inteiros. Este artigo compara seu desempenho em três cenários: simulação clássica sequencial, simulação clássica paralela com OpenMP e execução em hardware quântico da IBM. Os resultados mostram que a paralelização clássica alcança aceleração de até 7,5x, enquanto os processadores quânticos atuais ainda sofrem com ruído e decoerência. O estudo destaca a distância entre o potencial teórico do algoritmo e sua viabilidade prática, apontando os desafios tecnológicos a superar.*

1. Introdução

Grande parte da infraestrutura digital global depende de criptografia de chave pública, com destaque para o algoritmo Rivest-Shamir-Adleman (RSA), amplamente usado na proteção de dados sensíveis [Rivest et al. 1978]. Sua segurança baseia-se na dificuldade de fatorar inteiros grandes, motivo pelo qual permanece em uso há mais de 40 anos.

A principal ameaça ao RSA é o algoritmo de Shor, que, em um computador quântico com correção de erros, poderia fatorar inteiros em tempo polinomial [Shor 1994], tornando obsoletos os atuais criptossistemas de chave pública [Moorhead 2024, Goodin 2023]. Contudo, há uma grande lacuna entre sua promessa teórica e sua execução prática em arquiteturas clássicas e quânticas emergentes.

Este artigo analisa essa lacuna por meio de três abordagens: implementação clássica sequencial, paralela com OpenMP e execução em hardware quântico. A principal contribuição é a análise comparativa do desempenho da paralelização via OpenMP

em relação à linha de base sequencial e à execução quântica, evidenciando limites e potencialidades de cada cenário. Apesar dos avanços em iniciativas para migrar sistemas para algoritmos pós-quânticos aprovados pelo *National Institute of Standards and Technology* (NIST), compreender o desempenho do algoritmo de Shor segue essencial para antecipar riscos e orientar migrações seguras [Krelina 2021, Vianna and Camelo 2020].

O restante do artigo organiza-se da seguinte forma: Seção 2 discute trabalhos relacionados; Seção 3 revisa os fundamentos do algoritmo; Seção 4 descreve as metodologias; Seção 5 apresenta os resultados; e Seção 6 traz as conclusões.

2. Trabalhos Relacionados

A implementação do algoritmo de Shor é um tema recorrente na literatura, tanto em simulações clássicas quanto em execuções em hardware quântico. Simulações em computação de alto desempenho (*high performance computing* - HPC) são utilizadas para investigar o comportamento do algoritmo para um número de qubits que excede a capacidade dos dispositivos quânticos atuais. Por exemplo, trabalhos como o de [Smolin et al. 2013] exploram otimizações em simulações clássicas para testar os limites da fatoração. Outros estudos focam na paralelização da simulação de circuitos quânticos, demonstrado em [Häner and Steiger 2016], que apresenta técnicas para simular um grande número de qubits em supercomputadores.

No que tange à execução em *hardware* quântico, a fatoração do número 15 tornou-se um experimento canônico. Um dos primeiros relatos foi o de [Vandersypen et al. 2001], utilizando uma molécula com 7 qubits em um sistema de Ressonância Magnética Nuclear (RMN). Implementações mais recentes em plataformas de qubits supercondutores, como as da IBM, são exploradas em [Gidney and Ekerå 2021], que propõe otimizações para reduzir drasticamente o número de qubits necessários. Nosso trabalho se diferencia por realizar uma comparação direta e sistemática do desempenho entre uma implementação paralela otimizada com OpenMP e a execução em processadores quânticos de última geração, fornecendo uma perspectiva prática sobre a viabilidade de cada abordagem.

3. Algoritmo de Shor

Devido ao seu impacto sobre cifras assimétricas, especialmente RSA e curvas elípticas, o algoritmo de Shor tornou-se uma das aplicações mais relevantes da computação quântica [Krelina 2021]. Seu funcionamento baseia-se na identificação da periodicidade de funções modulares, as mesmas utilizadas em algoritmos como RSA. Tais funções apresentam natureza cíclica, com valores que se repetem sob determinada frequência. Identificar essa frequência é precisamente o ponto de vulnerabilidade explorado pelo algoritmo de Shor, descrito a seguir [Shor 1994, Nielsen and Chuang 2010].

Embora grande parte dos cálculos possa ser realizada em tempo polinomial por algoritmos clássicos, a determinação do período apresenta complexidade exponencial, tornando-se inviável para valores grandes de N. Nesse ponto, a computação quântica mostra-se especialmente promissora, pois resolve o problema do período em tempo polinomial [Shor 1994].

A estimação de fase quântica, implementada pela *Inversa da Transformada de*

Fourier Quântica ($QFT^\dagger$), permite determinar o período *r* em tempo polinomial, constituindo o principal fator da superioridade teórica do algoritmo de Shor.

Algorithm 1 Shor – Fatora um número inteiro N

Require: Inteiro N
1: Escolha um inteiro a tal que $1 < a < N$ e a seja coprimo com N
2: Encontre o período r de $f(x) = a^x \bmod N$
3: **if** r é par e $a^{r/2} \not\equiv -1 \pmod N$ **then**
4: Calcule $p = \gcd(a^{r/2} - 1, N)$ e $q = \gcd(a^{r/2} + 1, N)$
5: **return** Fatores primos p e q de N
6: **else**
7: Volte ao passo 1
8: **end if**

Na simulação, a $QFT^\dagger$ não é representada como uma matriz completa, mas decomposta em portas Hadamard, rotações de fase controladas $R_k^\dagger$ e trocas de qubits. Essa implementação algébrica, descrita em [Portugal 2024], reproduz o circuito e possibilita avaliar seu custo computacional na prática. No algoritmo apresentado acima, essa operação corresponde à segunda etapa.

4. Metodologia

4.1. Arquitetura Clássica

Na arquitetura clássica, implementamos duas versões do algoritmo de Shor sendo a primeira sequencial, focada na validação de resultados, e outra paralelizada com OpenMP, orientada a desempenho e escalabilidade em CPUs multicore. Ambas realizam simulação clássica do circuito quântico, reproduzindo as etapas essenciais, como a preparação do estado, aplicação das portas, oráculo modular e QFT/IQFT, por meio de representações matriciais/vetoriais e aritmética de alta precisão quando necessário. A versão sequencial facilita depuração e comparação com referências teóricas; já a versão paralela explora granularidade por laços e tarefas, reduzindo o tempo de execução para instâncias maiores, ao custo de maior complexidade de sincronização e consumo de memória. Em conjunto, as duas abordagens permitem validar corretude, medir speedups, analisar gargalos e estabelecer uma base reprodutível para estudos de otimização e portabilidade para outras plataformas.

4.1.1. Implementação Sequencial

A versão sequencial do algoritmo simula as operações quânticas utilizando apenas recursos computacionais clássicos. Para isto, as portas quânticas são representadas por matrizes, e a evolução do estado quântico é calculada por meio de produtos matriciais e produtos de Kronecker.

Essa implementação serve como base de comparação para as versões paralela e quântica, revelando os limites computacionais da abordagem clássica.

Algorithm 2 Shor – Implementação Clássica Simulando o Circuito Quântico

Require: Inteiro N
Ensure: Fatores primos de N
1: Selecione um valor a tal que $1 < a < N$ e $\gcd(a, N) = 1$
2: Defina o número de qubits do registrador de trabalho $n = \lceil \log_2 N \rceil$
3: Defina o número de qubits de contagem (e.g., $m = 2n$)
4: Inicialize o estado quântico $0^{\otimes(m+n)}$
5: Aplique portas Hadamard aos m qubits de controle
6: Aplique a exponenciação modular controlada: $U_f xy = xy \oplus a^x \bmod N$
7: Aplique a Transformada de Fourier Quântica Inversa ($QFT^\dagger$) aos qubits de controle
8: Meça os qubits de controle e obtenha o valor c
9: Estime o período r a partir de $c/2^m$ usando frações contínuas
10: **if** r é par e $a^{r/2} \not\equiv -1 \pmod N$ **then**
11: Calcule $p = \gcd(a^{r/2} - 1, N)$, $q = \gcd(a^{r/2} + 1, N)$
12: **return** Fatores p e q
13: **else**
14: Repita o processo com outro valor de a
15: **end if**

4.1.2. Implementação Paralela

A versão paralela busca reduzir o tempo de fatoração explorando duas camadas de paralelismo uma externa, testando diferentes candidatos a simultaneamente, e outra interna, dentro da simulação para cada candidato. São definidos dois parâmetros de controle: `TOTAL` (número total de *threads*); e `INTERNAL` (número de *threads* internas usadas em cada simulação).

São realizadas até 100 tentativas de fatoração por meio da geração de 100 candidatos a coprimos com N, distribuídos entre $TOTAL/INTERNAL$ *threads* externas. Cada uma dessas *threads* utiliza `INTERNAL` *threads* para paralelizar as operações matriciais da simulação. [1]

4.2. Implementação em Arquitetura Quântica

A implementação do componente quântico foi desenvolvida em Python com a biblioteca Qiskit. A execução foi realizada em processadores quânticos reais da IBM, permitindo a avaliação prática do algoritmo.

Para a fatoração de $N = 15$, o circuito foi construído com dois registradores principais. O *registrador de trabalho*, responsável pela exponenciação modular, requer $n = \lceil \log_2 15 \rceil = 4$ qubits. Para garantir precisão adequada na estimação de fase, o *registrador de contagem* foi implementado com $m = 2n = 8$ qubits. Portanto, o circuito total utilizou 12 qubits, seguindo a construção teórica padrão [Nielsen and Chuang 2010].

Para avaliação experimental, o circuito foi transpilado com nível de otimização 1 e executado em dois backends distintos da IBM Quantum: o `ibm_torino` (processador Heron r1, 133 qubits) e o `ibm_sherbrooke` (processador Eagle r3, 127 qubits).

[1] Um diagrama esquemático está disponível externamente em: github.com/EvelynHenriques/pfc-shor.

Cada circuito foi executado com 4096 amostragens (*shots*) para obter uma distribuição de probabilidade das medições.

5. Resultados e Discussão

5.1. Avaliação na Arquitetura Clássica

Os experimentos em ambiente clássico foram conduzidos em nós computacionais do cluster do Instituto Militar de Engenharia. Cada nó utilizado é equipado com 80 núcleos de CPU a 2,10 GHz, 187,35 GB de memória RAM e opera com o sistema operacional Linux (kernel 3.10.0-693.5.2.el7.x86_64) sobre a arquitetura x86_64. O objetivo foi otimizar o uso de paralelismo e avaliar o ganho de desempenho.

Configuração de Paralelismo e Escalabilidade

O primeiro experimento buscou a melhor configuração entre paralelismo externo e interno. Foram realizadas 2000 tentativas de fatoração para inteiros $N = p \cdot q$, com $p \neq q$ e $N \leq 203$. A análise mostrou que a configuração mais eficiente ocorria quando `INTERNAL` = `TOTAL`/2.

Com base nesse resultado, um segundo experimento analisou o *speedup*. O teste utilizou os inteiros $N = 85$, 123 e 219, com variação no número total de *threads* de 2 até 80. A Figura 2 mostra que o ganho cresce de forma quase linear até cerca de 24 *threads*. A partir daí, os benefícios tornam-se marginais. O maior *speedup* registrado foi de aproximadamente $7{,}5\times$. A ausência de escalonamento linear até 80 *threads* é atribuída as porções sequenciais inerentes ao algoritmo (Lei de Amdahl), que limitam os ganhos da paralelização.

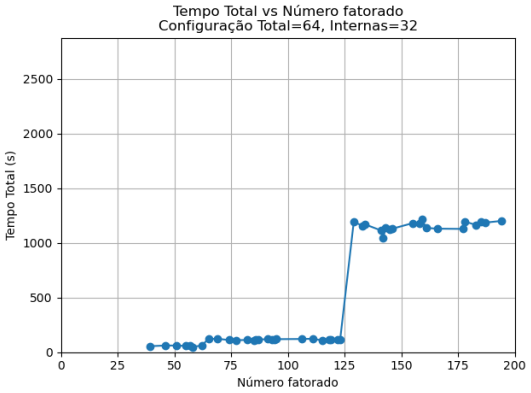

Figura 1. Tempo de execução para `TOTAL = 64`, `INTERNAL = 32`.

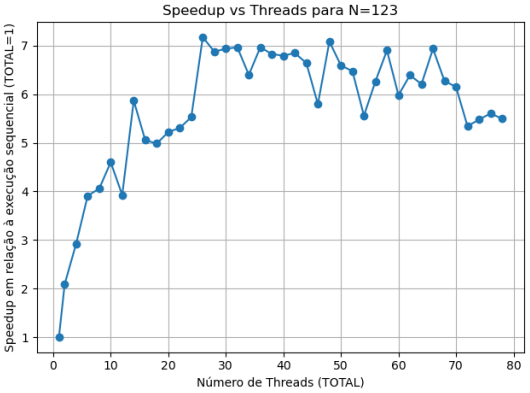

Figura 2. Gráfico de Speedup vs. número de *threads* para $N = 123$.

Comparação entre Versões Sequencial e Paralela

A comparação entre as implementações sequencial e paralela (`TOTAL = 64`, `INTERNAL = 32`) confirmou a efetividade do paralelismo. Como ilustrado na Figura 3,

o tempo de execução exibe um padrão de “degraus”, que ocorrem quando um qubit adicional é necessário para representar N, elevando exponencialmente o custo da simulação (dobrando o tamanho do vetor de estado).

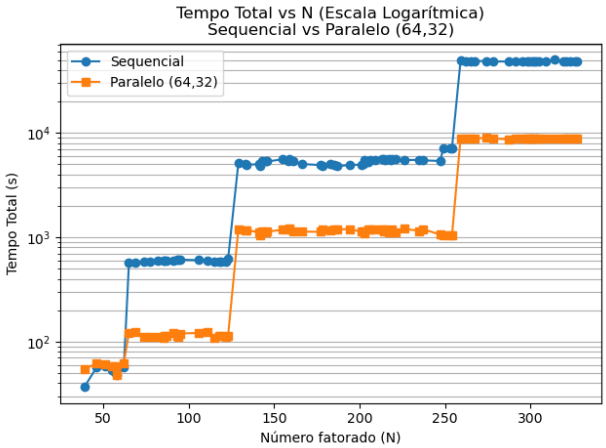

Figura 3. Tempo total (escala log) para as versões sequencial e paralela.

5.2. Avaliação em Arquitetura Quântica

A execução do algoritmo para fatorar $N = 15$ com base $a = 2$ nos processadores quânticos da IBM produziu resultados distintos.

Resultados Experimentais e Análise Crítica

A Figura 4 apresenta os resultados do `ibm_torino`. A distribuição de probabilidade mostra picos nos estados correspondentes à resposta correta, que se destacam significativamente do ruído de fundo. Em contraste, no `ibm_sherbrooke` (Figura 5), o resultado apresentou uma distribuição muito mais ruidosa, onde os estados corretos não são facilmente distinguíveis do ruído, tornando a extração do resultado correto impraticável sem conhecimento prévio.

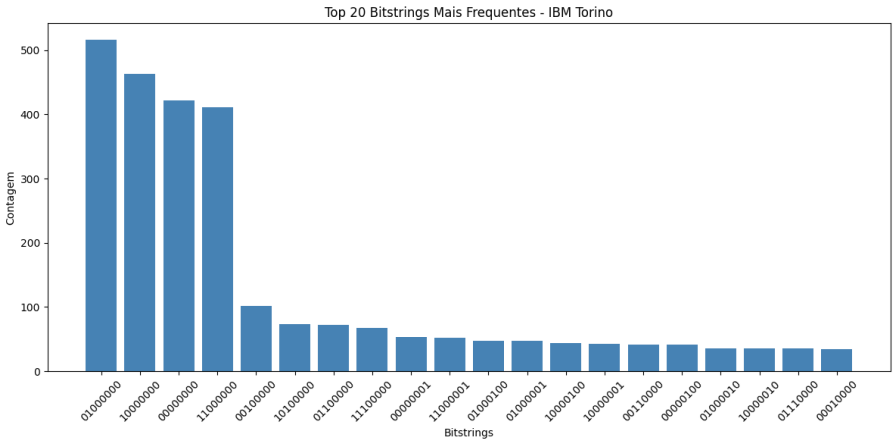

Figura 4. Resultados no `ibm_torino`: picos de probabilidade distinguíveis.

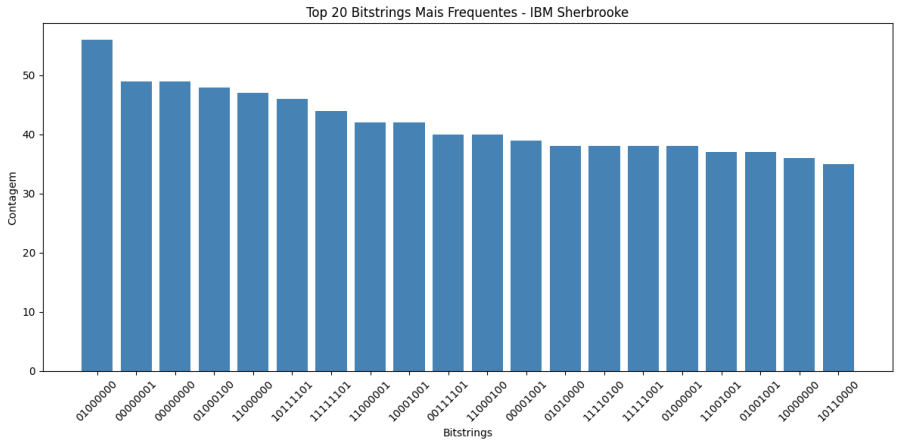

Figura 5. Resultados no `ibm_sherbrooke`: resposta correta ofuscada pelo ruído.

A diferença de qualidade decorre da profundidade do circuito transpilado: 3892 camadas no `ibm_sherbrooke` contra 3343 no `ibm_torino` (16% menor). Sem correção de erros, circuitos mais profundos aumentam o tempo de execução e a exposição dos qubits à decoerência e ao ruído das portas, degradando o resultado.

A Tabela 1 apresenta os tempos de processamento informados pela API da IBM. Esses valores não correspondem à execução física de cada circuito (na escala de microssegundos), mas ao tempo total de serviço — sem incluir a fila — para adquirir as 4096 amostragens (*shots*). O estado quântico precisa ser preservado apenas durante cada execução individual, que é muito rápida. O tempo total de 3 a 5 segundos é dominado pelo overhead do sistema de controle clássico (reset, execução e leitura), explicando a diferença entre a curta duração dos fenômenos quânticos (menores que 1 ms) e o tempo reportado.

Tabela 1. Tempos de processamento na API da IBM Quantum.

Backend	Transpilação (s)	Tempo Total da Tarefa (s)
`ibm_sherbrooke`	1,03	5,00
`ibm_torino`	1,18	3,00

6. Conclusão

Este trabalho comparou o desempenho do algoritmo de Shor em plataformas clássicas (paralela) e quânticas, revelando que a versão paralela obteve um speedup de até 7,5x, o que ressalta a eficácia da computação de alto desempenho (HPC) para simular algoritmos quânticos.

A abordagem quântica, por outro lado, expôs os desafios concretos das plataformas atuais. Embora a teoria prometa vantagens exponenciais, as limitações impostas pela alta taxa de erro, decoerência e conectividade limitada dos qubits tornam inviável, no presente momento, uma competição direta com implementações clássicas otimizadas para problemas de pequena escala.

Apesar das limitações, a experimentação confirma o potencial disruptivo da computação quântica. O fundamento teórico do algoritmo de Shor, que resolve em tempo polinomial um problema classicamente intratável, permanece sólido [Shor 1994]. A lacuna de desempenho decorre da imaturidade do hardware; com avanços em correção de erros e qualidade dos qubits, espera-se que algoritmos como o de Shor redefinam a segurança criptográfica.

Referências

Gidney, C. and Ekerå, M. (2021). How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433.

Goodin, D. (2023). Breaking RSA encryption just got 20x easier for quantum computers. *CSO Online*.

Häner, T. and Steiger, D. S. (2016). High performance emulation of quantum circuits. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16, page 87, Piscataway, NJ, USA. IEEE Press.

Krelina, M. (2021). Quantum technology for military applications. *EPJ Quantum Technology*, 8.

Moorhead, P. (2024). IBM prepares for a quantum-safe future using crypto-agility. Accessed on 22 July 2025.

Nielsen, M. A. and Chuang, I. L. (2010). *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA.

Portugal, R. (2024). Basic Quantum Algorithms. Acesso em mar 2025 e disponível em: `https://arxiv.org/abs/2201.10574`.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE.

Smolin, J. A., Smith, G., and Vargo, A. (2013). Classical simulation of the quantum fourier transform with an unentangled product state. *Physical Review A*, 87:032306.

Vandersypen, L. M., Steffen, M., Breyta, G., Yannoni, C. S., Sherwood, M. H., and Chuang, I. L. (2001). Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414(6866):883–887.

Vianna, E. W. and Camelo, J. R. S. (2020). Defesa cibernética no Brasil: primícias de uma história de sucesso. *Revista da Escola Superior de Guerra*, 35(75):127–154.