# Análise de desempenho de linguagens de programação e bibliotecas quânticas

**Jorge G. Viegas**[1], **Gilberto Irajá Müller**[1]

[1]Universidade do Vale do Rio dos Sinos - UNISINOS
Caixa Postal 93.022-718 – São Leopoldo – RS – Brasil

jorgematheusv@gmail.com, gimuller@unisinos.br

***Abstract.*** *In recent years, several software tools such as programming languages, libraries and simulators were developed to help the development of new quantum algorithms, making it difficult for students and scientists to choose the quantum software to be used. This paper presents a performance analysis of quantum programming languages and libraries using quantum algorithms present in the literature. To achieve the objectives of this paper, an experimental research was carried out using the stopwatch technique for data collection through a quantitative approach. Finally, the analysis and discussion of results, final considerations and proposals for future work are presented.*

***Resumo.*** *Nos últimos anos, diversas ferramentas de software como linguagens de programação, bibliotecas e simuladores foram desenvolvidas para auxiliar o desenvolvimento de novos algoritmos quânticos, tornando difícil a escolha de estudantes e cientistas sobre o software quântico a ser utilizado. O presente artigo apresenta uma análise de desempenho de linguagens de programação e bibliotecas quânticas utilizando algoritmos quânticos presentes na literatura. Para atingir os objetivos deste artigo, foi realizada uma pesquisa experimental utilizando a técnica stopwatch para a coleta de dados através de uma abordagem quantitativa. Por fim, apresentam-se a análise e discussão de resultados, considerações finais e propostas de trabalhos futuros.*

## 1. Introdução

Em 1982, Richard Philips Feynman sugeriu que um computador que utilizasse propriedades da mecânica quântica poderia habilitar simulações eficientes de sistemas físicos e quânticos, pois seria impossível para um computador universal clássico representar resultados de sistemas quânticos [Feynman 1982]. Segundo [Arute 2019], as observações de Feymann impuseram o início dos estudos sobre a computação quântica.

Através da utilização de propriedades da mecânica quântica, como emaranhamento e superposição, computadores quânticos são capazes de realizar tarefas específicas exponencialmente mais rápidas do que os computadores clássicos [Zhao 2020]. Para [Svore and Troyer 2016], o grande poder do computador quântico está na capacidade de aplicar as propriedades da mecânica quântica para solucionar problemas computacionais clássicos como, por exemplo, criptografia, simulação de sistemas físicos e solução de sistemas lineares. Os algoritmos quânticos são algoritmos que se utilizam das propriedades quânticas como o emaranhamento e superposição através de uma combinação de

operações e transformações em *qubits*, representadas por um circuito quântico para resolver um problema computacional. A Tabela 1 apresenta os principais algoritmos quânticos presentes na literatura.

**Tabela 1. Algoritmos quânticos constantes na literatura**

| Algoritmo | Descrição |
|---|---|
| Bernstein-Vazirani | Dada uma função $f(x) : \{0,1\}^n \rightarrow \{0,1\}$ que tem como propriedade sempre retornar o resultado do produto *bitwise* entre $x$ e uma sequência de *bits* $a$, o algoritmo encontra a sequência de *bits* $a$ [Bernstein and Vazirani 1997]. |
| Deutsch-Jozsa | Dada uma função booleana $f(x) : \{0,1\}^n \rightarrow \{0,1\}$, garantida como balanceada ou constante, o algoritmo descobre a propriedade da função em apenas uma execução [Deutsch and Jozsa 1992]. |
| Grover | Dada uma lista de $n$ itens não-ordenados, o problema se dá em encontrar um item que satisfaça uma propriedade única dentre os itens da lista. Pode ser definido através de uma função $f(x) : \{0,1\}^n \rightarrow \{0,1\}$, que retorne 1 para apenas um valor de $x$. O algoritmo busca o valor do parâmetro $x$ que retorne o valor 1 [Grover 1997]. |
| Shor | Utilizado na fatoração de números inteiros, o algoritmo encontra, dados dois números inteiros $a$ e $N$, o período de uma função $f(x) = a^r \mod N$ [Shor 1994]. |
| Simon | Dada uma função $f(x) : \{0,1\}^n \rightarrow \{0,1\}^m$, o algoritmo encontra a sequência de *bits* que defina a relação de invariância $XOR$ da função dois-para-um para os dois valores de entrada ($x$ e $x'$) que retornam o mesmo valor de $y$ [Simon 1997]. |

Segundo [Svore and Troyer 2016], o desenvolvimento e estudo de linguagens de programação e bibliotecas quânticas acelera o desenvolvimento de novos algoritmos quânticos. A pesquisa na área das linguagens de programação quânticas está em evolução e espera-se que diversas linguagens sejam desenvolvidas antes mesmo da disseminação do computador quântico [Garhwal et al. 2019]. Apesar do reflexo positivo desse crescimento, aumenta-se a dificuldade de estudantes e pesquisadores decidirem qual linguagem utilizar, se perdendo na documentação ou simplesmente não sabendo por onde começar [LaRose 2019].

Com base no contexto apresentado, esse artigo pode ser representado pela seguinte questão de pesquisa: **Qual o panorama das linguagens de programação e bibliotecas quânticas ao considerar o desempenho na implementação de algoritmos quânticos recorrentes na literatura?** Para responder essa questão de pesquisa, os seguintes objetivos foram realizados: (a) implementação de algoritmos quânticos constantes na literatura de acordo com linguagens de programação e bibliotecas quânticas; (b) análise quantitativa de desempenho de linguagens de programação e bibliotecas quânticas; e (c) um repositório no Github com os algoritmos quânticos utilizados neste artigo.

Além da introdução, o artigo possui a seguinte organização: a Seção 2 apresenta os trabalhos relacionados. Na Seção 3 são mostrados os resultados e discussão dos experimentos. Por fim, a Seção 4 conclui o artigo apresentando os trabalhos futuros.

## 2. Trabalhos Relacionados

Há diversos trabalhos sobre linguagens de programação quânticas com foco em características e funcionalidades. A seguir, apresentam-se os principais trabalhos correlatos.

Em [LaRose 2019], o autor realizou uma pesquisa sobre diferentes ambientes de computação quântica, com o objetivo de prover uma visão geral das plataformas de software quântico disponíveis. A pesquisa compara as plataformas em diferentes aspectos, tais como: instalação, documentação e tutoriais, sintaxe das linguagens, suporte das bibliotecas, hardware quântico e compiladores quânticos. As plataformas consideradas na pesquisa são: pyQuil, Qiskit, ProjectQ e Quantum Development Kit. [LaRose 2019] também avaliou o desempenho dos simuladores locais das plataformas Qiskit e ProjectQ e dos hardwares quânticos disponíveis das plataformas pyQuil e Qiskit, utilizando algoritmos de profundidade.

Em [Garhwal et al. 2019], os autores realizaram uma pesquisa com o objetivo de comparar linguagens de programação quânticas de alto nível, levando em consideração suas características e funcionalidades. A pesquisa tem como objetivo a comparação de linguagens implementadas em hardware quântico, não abordando simuladores e emuladores quânticos. A pesquisa abordou como questões centrais as diferentes linguagens de programação quânticas, as tendências na área de pesquisa de software quântico e a comparação e classificação de linguagens em paradigmas de programação.

Em [Heim et al. 2020], os autores realizaram um estudo sobre linguagens de programação quânticas, destacando aspectos importantes da programação quântica e como ela difere da programação clássica. O estudo objetiva apresentar o estado da arte na área das linguagens de programação quânticas, destacando suas principais características e provendo exemplos de código fonte. A pesquisa tem como motivação a crescente relevância do estudo das linguagens de programação quânticas e sua importância na arquitetura do modelo computacional quântico. Os autores destacam casos de uso relevantes de cada linguagem, demonstrando a importância das ferramentas de software como simuladores, bibliotecas, documentação e materiais de aprendizagem. As linguagens selecionadas seguindo uma abordagem qualitativa são: Q#, Qiskit, Cirq, Quipper e Scaffold.

A escolha das linguagens de programação e bibliotecas quânticas abordadas neste artigo considerou os seguintes critérios: (a) mais citadas nos trabalhos relacionados; (b) disponham de simuladores locais; (c) disponham de documentação; (d) relevância dos laboratórios em que a linguagem foi criada conforme [LaRose 2019]; e (e) possuam implementação dos algoritmos constantes na literatura. A partir desses critérios, as linguagens de programação e biliotecas quânticas abordadas são: Cirq, Pyquil, Q# e Qiskit.

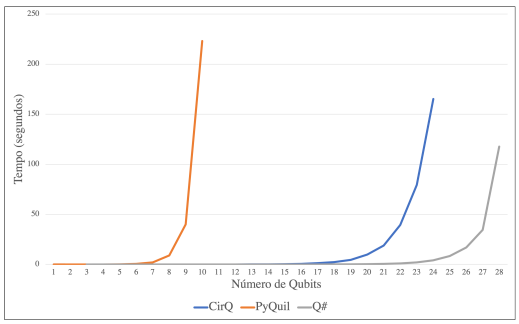
## 3. Resultados

Esta seção apresenta o método de coleta de dados, a análise e discussão dos resultados. Os experimentos foram executados em um MacBook Pro 2019, com memória DDR4 16GB 2400 MHz e processador Intel Core i7 2,6 GHz. Para a coleta de dados dos experimentos, o método *stopwatch* foi utilizado. O método consiste na utilização do relógio embutido no computador para estimar o tempo decorrido da execução de um programa [Stewart 2002]. Outliers foram identificados utilizando o método dos intervalos interquartis (Box-plot), que consiste em definir valores máximos e mínimos dentro da amostra. Observações situadas acima do máximo ou abaixo do mínimo foram consideradas *outliers* e desconsideradas na análise de resultados [Dawson 2011].

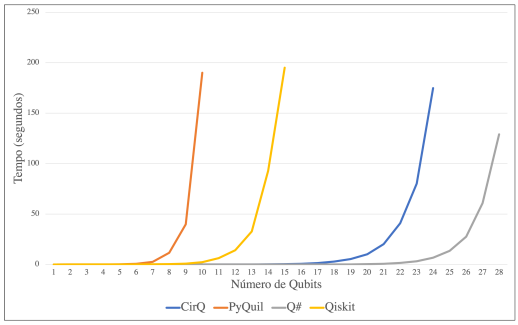
O cenário utilizado para o cálculo de estimativa do tamanho de amostra é a implementação do algoritmo de Deutsch-Jozsa na biblioteca Qiskit utilizando 10 *qubits*.

A escolha do cenário tem como base a relevância do algoritmo e da biblioteca nos trabalhos relacionados. A partir daí, foram realizadas 30 medições preliminares do cenário supracitado computando o tempo médio e o desvio padrão. O tamanho estimado de amostra foi de 42 medições, considerando um intervalo de confiança (IC) de 95% e margem de erro de 5% [Almeida 2014]. Ainda com base no tempo de execução obtido através de medições preliminares, foram definidos pontos de corte em função do tempo de processamento do *hardware* utilizado: (a) menor ou igual a 60 segundos, foram executadas as 42 medições; (b) maior que 60 segundos e menor que 300 segundos, foi executada uma única medição; e (c) maior que 300 segundos, a execução do programa foi interrompida e as medições não foram consideradas na análise.

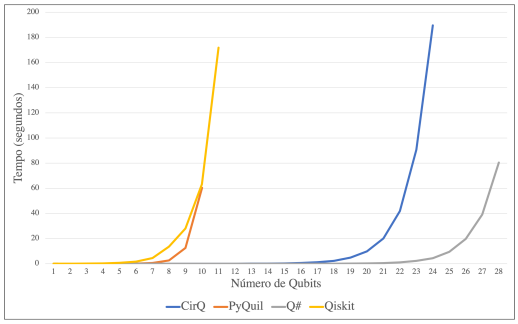
A Figura 1 apresenta os gráficos de desempenho dos algoritmos e linguagens de programação e bibliotecas quânticas.



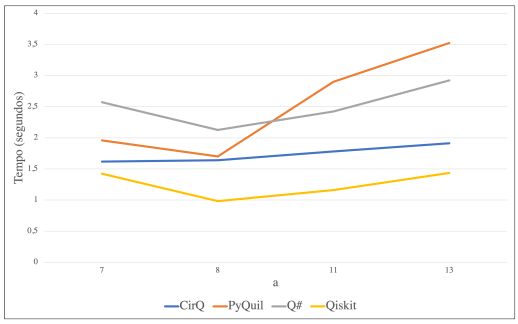

(a) Algoritmo de Bernstein-Vazirani



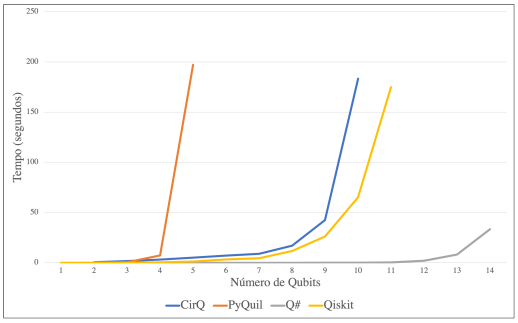

(b) Algoritmo de Deutsch-Jozsa




(c) Algoritmo de Grover




(d) Algoritmo de Shor




(e) Algoritmo de Simon

**Figura 1. Gráficos de desempenho dos algoritmos**

Ressalta-se que o algoritmo de Bernstein-Vazirani na biblioteca Qiskit não apre-

sentou resultados consistentes, pois o tempo de execução se manteve constante independentemente do número de *qubits*, comportamento que viola a teoria da representação de *qubits* em estruturas de memória clássicas. Por este motivo, o cenário foi desconsiderado da análise. Ainda, o Algoritmo de Shor (Figura 1(d)) tem dois números inteiros como parâmetros e não varia o número de *qubits*. Para a análise, foi utilizado o número $N = 15$ variando o parâmetro $a$, mostrando que o Qiskit teve o menor tempo.

As Figuras 1(a), 1(b), 1(c) e 1(e) apresentam comportamento assintótico exponencial em relação ao número de *qubits*. Pelo fato dos *qubits* serem capazes de entrar em um estado de superposição, os simuladores locais também necessitam representar estes estados quânticos em estruturas de dados clássicas. Para representar $n$ *qubits* em estruturas de memória clássicas, são necessários $2^n$ bits, mostrando assim o comportamento exponencial. Ao aplicar as operações quânticas para alterar o estado dos *qubits*, diversas operações de produto tensorial (espaço vetorial) são aplicadas à todas as combinações de estado possíveis, resultando na complexidade $\mathcal{O}(2^n)$. Por este motivo, existe uma limitação dos simuladores locais em relação ao número de *qubits* [Nielsen and Chuang 2010].

A linguagem Q# obteve, no geral, o melhor desempenho dentre as linguagens de programação e bibliotecas quânticas abordadas. O simulador local da linguagem Q# executou cenários com até 28 *qubits*, sendo a linguagem com o simulador local de maior capacidade. O simulador local da linguagem Cirq também obteve desempenho significativo em relação às outras linguagens, sendo possível executar cenários com até 24 *qubits*. A Tabela 2 apresenta o número máximo de *qubits* obtido em cada algoritmo.

**Tabela 2. Número máximo de *qubits***

| Algoritmo | Cirq | Pyquil | Q# | Qiskit |
|---|---|---|---|---|
| Bernstein-Vazirani | 24 | 10 | 28 | - |
| Deutsch-Jozsa | 24 | 10 | 28 | 15 |
| Grover | 24 | 10 | 28 | 12 |
| Simon | 10 | 5 | 14 | 12 |

## 4. Conclusões e Trabalhos Futuros

Este artigo abordou a análise de desempenho de linguagens de programação e bibliotecas quânticas utilizando algoritmos quânticos recorrentes na literatura, motivado pela dificuldade de estudantes e pesquisadores decidirem qual linguagem utilizar. A pesquisa de trabalhos relacionados apresentou três trabalhos correlatos, que auxiliaram na definição das linguagens e bibliotecas abordadas na análise de desempenho. A coleta de dados considerou diferentes cenários de análise de desempenho, tais como: algoritmos, linguagens de programação, bibliotecas e o número de *qubits*. Os resultados indicam que a linguagem Q# obteve os melhores desempenhos dentre as linguagens abordadas, tanto em tempo médio como o número máximo de *qubits*. Para estudantes e pesquisadores com experiência prévia em desenvolvimento de software, sugere-se a utilização da linguagem Q#. Para estudantes e pesquisadores sem experiência prévia, a sugestão é a biblioteca Cirq. Por fim, foi disponibilizado um repositório no *Github*[1] contendo a implementação dos diferentes algoritmos nas linguagens de programação e bibliotecas quânticas abordadas neste artigo com o objetivo de contribuir com a comunidade de computação quântica.

[1] Acesse o repositório em: https://github.com/jorgeviegas/quantumalgorithms

Como trabalhos futuros, sugerem-se: (a) proposição de uma taxonomia de linguagens de programação e bibliotecas quânticas; e (b) análise quantitativa de desempenho de ambientes computacionais de hardware quântico disponíveis através de computação em nuvem, tais como: Microsoft Azure Quantum, IBM Q Experience e Google Quantum Computing Service.

## Referências


Almeida, V. (2014). Métodos quantitativos para ciência da computação experimental.

Arute, F. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510.

Bernstein, E. and Vazirani, U. (1997). Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473.

Dawson, R. (2011). How significant is a boxplot outlier? *Journal of Statistics Education*, 19(2):null.

Deutsch, D. and Jozsa, R. (1992). Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558.

Feynman, R. P. (1982). Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488.

Garhwal, S., Ghorani, M., and Ahmad, A. (2019). Quantum programming language: A systematic review of research topic and top cited languages. *Archives of Computational Methods in Engineering*.

Grover, L. K. (1997). Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 79(2):325–328.

Heim, B., Soeken, M., Marshall, S., Granade, C., Roetteler, M., Geller, A., Troyer, M., and Svore, K. (2020). Quantum programming languages. *Nature Reviews Physics*, 2(12):709–722.

LaRose, R. (2019). Overview and comparison of gate level quantum software platforms. *Quantum*, 3:130.

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

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

Simon, D. (1997). On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483. cited By 416.

Stewart, D. B. (2002). Measuring execution time and real-time performance. In *In: Proceedings of the Embedded Systems Conference (ESC SF*, pages 1–15.

Svore, K. M. and Troyer, M. (2016). The quantum future of computation. *Computer*, 49(9):21–30.

Zhao, J. (2020). Quantum software engineering: Landscapes and horizons.