ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 10, No. 1, 2025

QUANTUM-INSPIRED COMPUTING: COMPARSION OF VARIANTS OF SHOR'S ALGORITHM

Volodymyr Pavlenko, Valerii Hlukhov

Lviv Polytechnic National University, 12, Bandera Str, Lviv, 79013, Ukraine
Authors' e-mails: *volodymyr.v.pavlenko@lpnu.ua, valerii.s.hlukhov@lpnu.ua*

https://doi.org/10.23939/acps2025.01.071

Submitted on 16.03.2025

***Abstract*: This study explores the theme of quantum-inspired computing, specifically the different variations of Shor's algorithm. The focus of this article is on leveraging quantum computing's approach to explore new ways to solve complex problems more efficiently than classical methods. Using the Microsoft Azure Quantum SDK, we have simulated variant of Shor's algorithm to investigate its effectiveness in solving complex problems more efficiently than traditional methods. Although variant has demonstrated good potential for translating quantum principles to classical algorithms, it is not practical in terms of efficiency or scalability. It is relatively slow, highlighting its limitations in application. Nevertheless, it offers a valuable example of quantum-inspired algorithm design by reducing quantum complexity and introducing novel classical approaches.**

***Index terms*: Shor's algorithm, Microsoft Azure Quantum SDK, Quantum-inspired algorithms, Quantum-inspired cryptography, Quantum-inspired efficiency**[1]

I. INTRODUCTION

Quantum computing represents a significant leap forward in our ability to solve complex problems that classical computers struggle with. At the heart of this advancement is Shor's algorithm [1], renowned for its ability to factor large numbers efficiently – a process critical to cryptography and secure digital communication.

Recent advancements in quantum-inspired computing signify a pivotal shift in computational capabilities, drawing from the principles of quantum mechanics to enhance classical computing frameworks [2]. Notably, quantum-inspired algorithms have demonstrated substantial improvements in machine learning efficiency, facilitating faster data processing and analysis [3]. These advancements extend to practical applications such as optimization problems in global financial markets and the potential enhancement of future 6G communication networks [4].

Furthermore, the application of quantum-inspired methods in drug discovery and material science promises to accelerate the development of new therapeutics and materials through more effective molecular simulations [5].

The variation of Shor's algorithm [6] is a nuanced adaptation that explores a new approach to the usage of quantum Shor's algorithms in quantum-inspired systems [2]. Variation introduces a quantum-inspired methodology that enhances the efficiency of prime factorization, a key operation in encryption and digital security.

II. LITERATURE REVIEW AND PROBLEM STATEMENT

Traffic Optimization: genetic algorithms have shown promise in addressing complex path-finding and scheduling challenges in traffic flow. Quantum-inspired genetic algorithms, which integrate principles from quantum computing to enhance search and optimization capabilities, have been successfully applied in various scenarios. For instance, Narayanan and Moore demonstrated the potential of quantum-inspired genetic algorithms to improve evolutionary computation techniques [7]. Similarly, Wang and Li developed a hybrid quantum-inspired genetic algorithm specifically designed for flow shop scheduling, illustrating its effectiveness in managing combinatorial optimization problems, which could be applicable to optimizing traffic flow paths [8].

Supply Chain Management: Quantum-inspired algorithms have been utilized to optimize supply chain operations, including logistics and warehouse management [9]. Companies like BMW have explored quantum-inspired solutions to solve complex logistics problems, such as the routes optimization of their supply chain to minimize costs and improve efficiency [10].

Portfolio Optimization: Quantum-inspired computing has been applied in the finance sector to optimize investment portfolios, balancing risk and return in a way that is computationally efficient and provides better outcomes than classical algorithms. Companies like Barclays have explored quantum-inspired techniques to enhance their portfolio optimization processes [11].

Molecular Simulation: Quantum-inspired algorithms are used to simulate the properties of materials [5] at the molecular level, which is essential for discovering new materials and drugs. These simulations can predict material behaviors under various conditions, helping in the design of

[1] The article uses materials and results obtained by the authors during the research project "Intelligent Methods and Tools for Designing Modules for Autonomous Cyber-Physical Systems" state registration number 0124U002340 dated 09.03.2024. This project is conducted at the Department of Electronic Computing Machines of the Institute of Computer Technologies, Automation, and Metrology of the Lviv Polytechnic National University from 2024 to 2028.

new drugs or materials with desired properties more efficiently than traditional methods.

Feature Selection: Quantum-inspired computing has been used to enhance machine learning models by optimizing the feature selection process [12]. This involves selecting the most relevant features from large datasets to improve the accuracy of machine learning models while reducing computational complexity.

Data Clustering: Quantum-inspired algorithms have been applied to data clustering tasks, such as customer segmentation in marketing. These algorithms can find the optimal grouping of data points more efficiently than classical methods, enabling businesses to target their marketing strategies more effectively [10].

Power Grid Optimization: In the energy sector, quantum-inspired computing has been used to optimize the operation of power grids [13], including the efficient distribution of renewable energy sources. This helps minimize energy loss and ensures the stability of the power supply across the grid.

These examples illustrate the versatility and potential of quantum-inspired computing across various industries. By applying principles of quantum mechanics to classical computing, quantum-inspired technologies provide solutions to problems that were previously intractable or highly resource-intensive [10].

III. SCOPE OF WORK AND OBJECTIVES

The purpose of this work is to explore the potential for simplifying the quantum components of quantum algorithms, leveraging the Microsoft Azure Quantum SDK and Q# simulation of the variant of Shor's algorithm [6]. By focusing on reducing the complexity of quantum subroutines, this study investigates the feasibility of creating quantum-inspired algorithms that retain essential functionality while improving accessibility and practical implementation.

This article demonstrates how simplifying quantum operations can inspire classical algorithm design, aiming to establish a proof-of-concept for translating quantum principles into classical computing. By implementing variant of Shor's algorithm [6], we seek to highlight the potential of such approaches for advancing quantum-inspired methodologies, particularly for complex problems like prime factorization.

Additionally, we touch on the potential role of Field-Programmable Gate Array (FPGA)-based platforms in executing these simplified algorithms. As indicated [6], there is a viable path for the application of variant [6] within the domain of quantum-inspired computing. This approach could leverage the adaptability and efficiency of digital quantum coprocessors, offering a practical framework for executing quantum algorithms in a more accessible manner.

IV. VARIANT OF SHOR'S ALGORITHM

The variant [6] significantly simplifies the quantum part of Shor's algorithm Fig. 1 by compiling the QFT part into 2 qubits subroutine. Original Shor's algorithm:

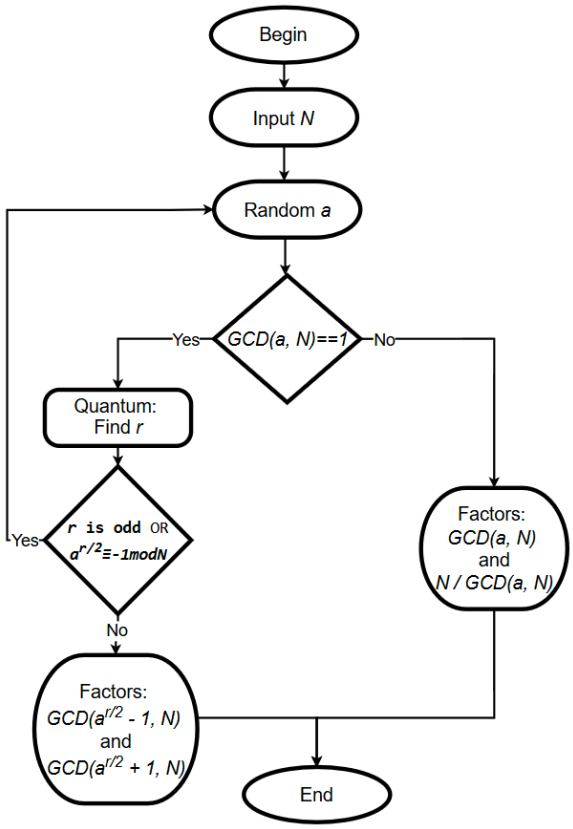

Fig. 1. Flow Chart of Shor's algorithm

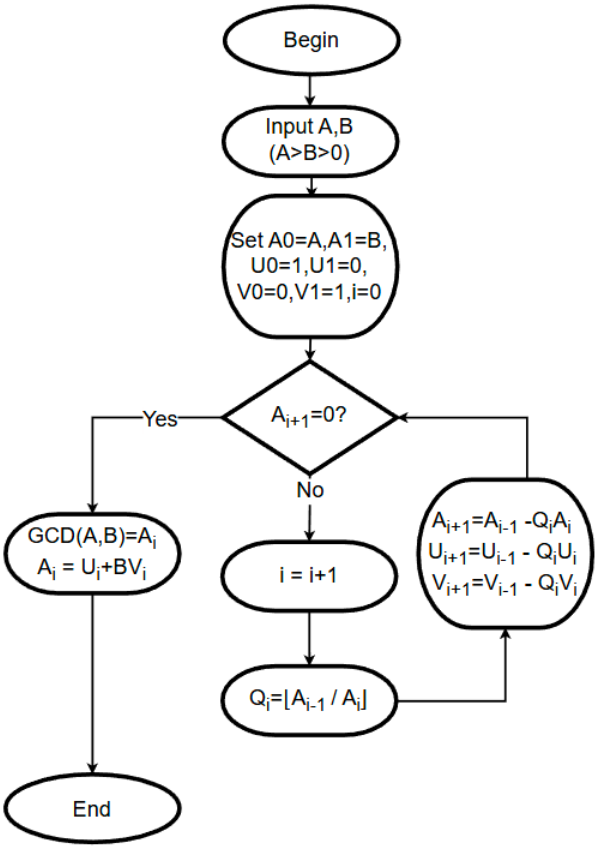

Fig. 2. Flow Chart of Extended Euclidean Algorithm

To factor an integer *N*, the following procedure is applied.

Firstly, a random integer *a* is selected such that it is less than *N* but greater than 1.

Next, the Greatest Common Divisor (GCD) of *a* and *N* is computed using Euclidean algorithm (Fig. 2). If the GCD is not 1, it is a non-trivial factor of *N*.

If the GCD is 1, a quantum state is prepared, and a Quantum Fourier Transform (QFT) is performed. In this variation [6], the period *r* is assumed to be $r = 2$.

After the QFT, the quantum state is measured. The measurement yields a value that is, with high probability, related to the period *r* of *a* modulo *N*.

Finally, post-processing is carried out. The GCD-based method [6] algorithm to extract factors of is used to derive factors of *N* from the measurement result. If no factors are obtained, the process is repeated with a new random *a*.

The flow diagram of Shor's algorithm outlines the key steps of executing Fig. 1.

The thing to understand about approach [6] is that the quadratic residue *a*, that it is getting us to compute, is the same *a* that Shor's algorithm is finding by using period-finding against a random base *g*. When we set $g = a$, we get an execution of Shor's algorithm where it computes that this g's period is 2, and returns $g^2 = a$ as the quadratic residue to use for factoring.

A sequence diagram of variation [6] is shown in Fig. 3.

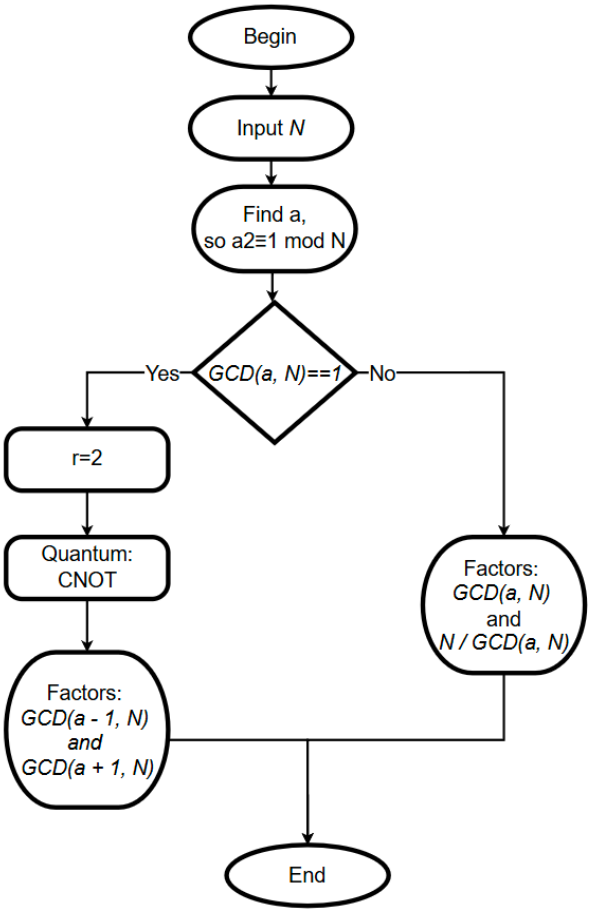

Fig. 3. Flow Chart of variant of Shor's algorithm

So, modification [6] (Fig. 3) focuses on optimizing the process by ensuring the period *r* is set to 2. It significantly simplifies the quantum circuit and reduce the overall resource requirements, such as the number of qubits. This adaptation allows for efficient implementation of Shor's algorithm on quantum-inspired computing platforms like FPGA quantum-inspired coprocessors [14].

A. PREPARING SIMULATION MODEL

For implementation, we are using Microsoft Azure Quantum SDK and language Q#. The program begins with setting up the value N. Then we perform the algorithm that is implemented by state machine (Fig. 3) mentioned above. Key steps include (the factorization below is done for test number 16837 generated by multiplying two prime numbers 113 and 149):

First step of this modification is to make a validation [6]. This part requires to use of Q# capabilities of simulation of qubits to generate coherent qubits, bring one of them in the state of a quantum superposition (using Hadamard gate), and after that perform CNOT (Controlled NOT) gate to provide their entanglement. The concrete quantum schema you can see in Fig. 4.

There is a base state $|+\rangle$ that can be described as $+ = \frac{0 + 1}{\sqrt{2}}$. In the circuit, the $|+\rangle$ acts as the control qubit input for controlled unitary operations, which are essential for creating the modular exponentiation steps in Shor's algorithm.

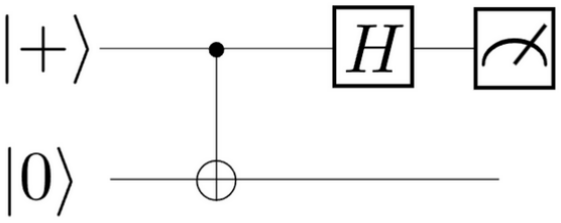

Fig. 4. The quantum circuit for compiled Shor's algorithm

It is important to know that the modular exponentiation step of Shor's algorithm in variant [6] is a CNOT gate and the Fourier transform part is a Hadamard gate [6] Q# code of the simulated operation:

```
operation ValidateAUsingQuantumSubroutine(a : Int, N :
Int, r : Int) : Result {
 use qubits = Qubit[2];

 // Prepare qubits
 H(qubits[0]);
 CNOT(qubits[0], qubits[1]);

 // Measurement could be used to validate assumptions
 let measurement = M(qubits[0]);

 ResetAll(qubits);
 return measurement;
}
```

The quantum aspect in approach [6] to factoring large numbers using a compiled variant [6] of Shor's

algorithm is largely for demonstration rather than practical necessity. It showcases how minimal quantum resources can still mimic Shor's algorithm [2].

The core of the algorithm involves a combination of classical and quantum steps, with the latter simulated for this demonstration. The quantum portion of the algorithm, while not executed on a quantum computer, is represented with quantum gates and measurements in a simulated environment. This approach allows for the exploration of the algorithm's principles and potential without the need for a physical quantum computing platform.

After the validation is complete, it's needed to have classical Post-Processing:

```
let factor = MaxI(
 GreatestCommonDivisorI(halfPower - 1, modulus),
 GreatestCommonDivisorI(halfPower + 1, modulus)
);

if (factor != 1) and (factor != modulus) {
 Message($"Found factor={factor}");
 return (true, (factor, modulus/factor));
}

return (false, (1, 1));
```

B. TIME AND SPACE COMPLEXITY

Let's evaluate the most significant parts of the created implementation and make an assumption about its time and space complexity using O(*n*) notation, where *n* – number of input bits and *N* – is an actual number.

Finding of a choosing a base a that satisfies $a^2 \equiv 1 mod N$ by computing modular inverses $\mathrm{p_q}$ and $\mathrm{q_p}$ using Extended GCD. The algorithm runs for $O\ \log \min\ a, N$ [6].

So, in terms of input bits count, using the Extended Euclidean Algorithm takes $O\ n^2$ time.

During finding *a* process GCD are done from 2 to N, so N-1 times. Iterating from 2 to N is $O\ 2^n$.

Overall routine complexity: $\mathrm{O}\ 2^n \cdot \mathrm{n}^2$.

GCD calculations after *a* is found: for a given *a*, calculate $\gcd\ \mathrm{a} - 1, \mathrm{N}$ and $\gcd\ \mathrm{a} + 1, \mathrm{N}$ using the Euclidean Algorithm; each GCD computation has a time complexity of $\mathrm{O}\ \mathrm{n}^2$; overall step complexity: $O\ n^2$.

The total time complexity combines the complexities of GCD, and iterations: $O\ 2^n \cdot n^2$. Best case complexity is based on the chance to find *a* on the first iteration, so $O\ n^2$.

This estimation (Table 1) indicates that variant [6] is not suitable for real-world usage due to its inefficiency and exponential time complexity for larger inputs.

According to classical implementation of Shor's algorithm (Table 2), in terms of bits *n* we can estimate that complexity is:

$$T_{\text{classical}} = O\ \ e^{c \cdot n^{1/3}\ \log n\ ^{2/3}}\ . \qquad (1)$$

So, rough estimation of time complexity for this variant will be $O\ \ 2^{n^{1/3}}$.

Table 1

Big O notation complexity

	Best case	Worst case
Time complexity	$O\ \mathrm{n}^2$	$\mathrm{O}\ 2^{\mathrm{n}} \cdot \mathrm{n}^2$
Space complexity	$\mathrm{O}(1)$	$\mathrm{O}(1)$

Table 2

Comparing variant and Classical Shor's algorithm implementation

	Variant [6] implementation	Classical simulation
Time complexity	$\mathrm{O}\ 2^{\mathrm{n}} \cdot \mathrm{n}^2$	$O\ 2^{n^{1/3}}$

When you simulate Shor's algorithm classically, there is no more efficient way to do it than computing every term of the wave function in the computational basis. So, we can see that variant [6] implemented in this research is worth than modern classical simulation of Shor's algorithm.

C. SIMULATION

The experiment will utilize three distinct semiprime numbers as test cases. Each number represents a different scale of challenge for the algorithm, ranging from relatively small numbers to larger integers that approach the sizes used in cryptographic applications.

The process involves selecting an appropriate auxiliary number *a* for each semiprime, which is crucial for the success of the algorithm. This choice is guided by the requirement that *a* and the semiprime share no common factors other than one, a condition that is necessary for the subsequent steps of the algorithm to yield meaningful results.

The semiprimes chosen for this experiment serve as test cases to evaluate the algorithm across a spectrum of complexities. These test cases are detailed in Table 3 and include three semiprime numbers of varying sizes:

- N1=1843=19*97,
- N2=16837=113*149,
- N3=20373811=5449*3739.
- The computation times have not been included in the Table 3 because the Q# and Microsoft Quantum Development Kit (QDK) tools utilized for this work do not provide functionality for precise time measurement. Furthermore, simulating quantum algorithms on classical hardware, as is the case with Q# and QDK, is not appropriate for factoring large numbers.
- The outcomes of the simulation highlight that, while variant of Shor's algorithm [6] is not practical for real-world applications due to its inefficiency and slow execution, it serves as an excellent example of how quantum algorithms can inspire classical approaches. For each test case, the algorithm successfully identifies a suitable a and deduces the prime factors of the

semiprime, demonstrating its conceptual validity and showcasing the potential of simplifying quantum elements to create innovative, quantum-inspired algorithms (Table 3).

Table 3

Simulation results

Number	Logs
N1	Message: *** Factoring 1843, attempt 1.
	Message: Starting searching needed 'a' for N=1843
	Message: Found 'a'=96
	Message: Assume that period=2.Message: Found factor=97
	Message: Found factorization 1843 = 97 * 19
	Result: "(97, 19)"
N2	Message: *** Factoring 16837, attempt 1.
	Message: Starting searching needed 'a' for N=16837
	Message: Found 'a'=6555
	Message: Assume that period=2.Message: Found factor=149Message: Found factorization 16837 = 149 * 113
	Result: "(149, 113)"
N3	Message: *** Factoring 20373811, attempt 1.
	Message: Starting searching needed 'a' for N=20373811
	Message: Found 'a'=9388628
	Message: Assume that period=2.
	Message: Found factor=5449
	Message: Found factorization 20373811 = 5449 * 3739
	Result: "(5449, 3739)"

This demonstrates that even limited or adapted versions of quantum algorithms can provide valuable insights, leading to the development of practical methods that leverage quantum concepts without requiring full quantum computation capabilities.

V. CONCLUSION

This study highlighted the transformative potential of quantum-inspired computing, particularly in fields like cryptography that rely on computational hardness assumptions, using variant [6] as a key example. While the simulations presented were simplified and failed to reflect the full complexity of quantum computation, they provided a valuable proof of concept for adapting quantum algorithms to classical problems.

The obtained results demonstrated algorithm's complexity equal $O\ n^2$ in the best cases and $O\ 2^n \cdot n^2$ for the worst cases. We can observe that variant [6] implemented in this research is worth than modern classical simulation of Shor's algorithm, but could be used for demonstration purposes to show how quantum algorithms could be transformed to theirs's classical variations.

While the practical application of the variant [6] was being limited by its high time complexity and scalability challenges, it successfully demonstrated how core concepts from quantum computing could influence classical algorithm design. Future research directions include optimizing the classical parts of the algorithm, exploring hardware accelerations via specialized architectures, and investigating more advanced quantum-inspired techniques that could close the gap between pure quantum and classical solutions.

Ultimately, the study confirms that even simplified models of quantum algorithms offer significant insights into the broader field of quantum-inspired computing, laying the groundwork for future innovations in cryptography, optimization, and beyond. By showcasing how quantum principles can be translated into classical operations, such approaches may foster the development of hybrid algorithms, promoting gradual integration of quantum computing concepts into practical applications before fully quantum hardware becomes widely available.

References

[1] Willsch, D., Willsch, M., Jin, F., De Raedt, H., and Michielsen, K., (2023). Large-Scale Simulation of Shor's Quantum Factoring Algorithm. *Mathematics*, 11(19), 4222. DOI: https://doi.org/10.3390/math11194222.

[2] Hlukhov, V., (2021). Implementation of Shor's Algorithm in a Digital Quantum Coprocessor. Proceedings of the 2nd International Conference on Intellectual Systems and Information Technologies (ISIT 2021), CEUR Workshop Proceedings, Vol. 3126, pp. 15–23. Available at: https://ceur-ws.org/Vol-3126/paper2.pdf.

[3] Huynh, L., Hong, J., Mian, A., Suzuki, H., Wu, Y., & Camtepe, S. (2023). Quantum-inspired machine learning: a survey. arXiv preprint arXiv:2308.11269. DOI: https://doi.org/10.48550/arXiv.2308.11269.

[4] Duong, T. Q., Ansere, J. A., Narottama, B., Sharma, V., Dobre, O. A., & Shin, H. (2022). Quantum-inspired machine learning for 6G: fundamentals, security, resource allocations, challenges, and future research directions. IEEE open journal of vehicular technology, 3, 375-387. DOI: https://doi.org/10.1109/OJVT.2022.3202876.

[5] Moussa, C., Wang, H., Araya-Polo, M., Bäck, T., & Dunjko, V. (2023, September). Application of quantum-inspired generative models to small molecular datasets. In 2023 *IEEE International Conference on Quantum Computing and Engineering (QCE)* (Vol. 1, pp. 342-348). IEEE. DOI: https://doi.org/10.48550/arXiv.2304.10867.

[6] Smolin, J. A., Smith, G., & Vargo, A. (2013). Pretending to factor large numbers on a quantum computer. *arXiv preprint arXiv:1301.7007*. DOI: https://doi.org/10.48550/arXiv.1301.7007

[7] Mansori, A. R., & Nguyeni, S. K. (2023). Quantum-Inspired Genetic Algorithms for Combinatorial Optimization Problems. *Algorithm Asynchronous*, 1(1), 16-23. DOI: https://doi.org/10.61963/jaa.v1i1.47

[8] Zhu, H., Luo, N. and Li, X., (2021). A Quantum-Inspired Cuckoo Co-Evolutionary Algorithm for No-Wait Flow Shop Scheduling. *IET Collaborative Intelligent Manufacturing*, 3(2), 105-118. DOI: https://doi.org/10.1049/cim2.12002

[9] Silveira, L. R., Tanscheit, R., & Vellasco, M. (2017). Quantum inspired evolutionary algorithm for ordering problems. *Expert Systems with Applications*, 67, 71–83. DOI: https://doi.org/10.1016/j.eswa.2016.08.067

[10] Arrazola, J. M., Delgado, A., Bardhan, B. R., & Lloyd, S. (2019). Quantum-inspired algorithms in practice. *arXiv*

preprint arXiv:1905.10415. DOI: https://doi.org/10.48550/arXiv.1905.10415

[11] Kuo, S. Y., Lai, Y. T., Jiang, Y. C., Chang, M. H., Wu, K. M., Chen, P. C., ... & Chou, Y. H. (2023, July). Entanglement Local Search-Assisted Quantum-Inspired Optimization for Portfolio Optimization in G20 Markets. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation (pp. 2232-2240).* DOI: https://doi.org/ 10.1145/3583133.3596370.

[12] Ram, P. K., Bhui, N., & Kuila, P. (2020, July). Gene selection from high dimensionality of data based on quantum inspired genetic algorithm. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1-5). IEEE. DOI: https://doi.org/10.1109/ICCCNT49239.2020.9225512

[13] Bertini, C., & Leporini, R. (2023). Quantum-inspired applications for classification problems. *Entropy, 25(3),* 404. DOI: https://doi.org/10.3390/e25030404

[14] Hlukhov, V. (2019). Implementing quantum Fourier transform in a digital quantum coprocessor. *Advances in Cyber-Physical Systems: scientific journal, 1 (4), 2019, 4(1),* 7-14. DOI: https://doi.org/10.23939/acps2019.01.006

[15] Fu, X.-Q., Bao, W.-S., Huang, H.-L., Li, T., Shi, J.-H., Wang, X., Zhang, S., and Li, F.-G., (2019). Realization of t-bit semiclassical quantum Fourier transform on IBM's quantum cloud computer. *Chinese Physics B*, 28, 2, 020302. DOI: https://doi.org/10.1088/1674-1056/28/2/020302

Volodymyr Pavlenko is a Computer Engineer with a strong experience in System Programming, who is currently working a Senior C++ developer. He completed his secondary education at Talne Lyceum of Maths and Economics from 2013 to 2017. Following this, he pursued higher education at Lviv Polytechnic National University earning a Bachelor's degree (2017-2021) and subsequently a Master's degree in Computer Engineering, specializing in System Programming. Currently, he is further advancing his expertise by working towards a Postgraduate degree in Computer Engineering at the same institution.

Valerii Hlukhov is a distinguished Doctor of Technical Sciences and Professor at the Department of Electronic Computing, Lviv Polytechnic National University. He pursued postgraduate studies at the Institute of Modeling Problems in Energy, Kyiv (1988-1991). He defended his candidate's thesis on mathematical special computers for navigation problems in 1991 and earned his Doctor of Technical Sciences degree in 2013. Dr. Hlukhov 's career includes extensive research and development in primary information processing devices and cryptoprocessors.