Hybrid Quantum Encoding: Combining Amplitude and Basis Encoding for Enhanced Data Storage and Processing in Quantum Computing

Bhattaraprot Bhabhatsatam
Faculty of Information Technology and Digital Innovation
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
s6007011966039@email.kmutnb.ac.th

Sucha Smanchat
Faculty of Information Technology and Digital Innovation
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
sucha.smanchat@itd.kmutnb.ac.th

2023 20th International Joint Conference on Computer Science and Software Engineering (JCSSE) | 979-8-3503-0050-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/JCSSE58229.2023.10201947

***Abstract*— This research applies Bit-Partition hybrid quantum encoding methods to store and process classical data in quantum systems efficiently. By combining amplitude encoding for representing the index and basis encoding for the data values, we introduce a novel technique that leverages the strengths of both encoding methods. We describe the process of encoding and decoding the hybrid states, highlighting the potential benefits of this approach in terms of data storage and computational efficiency. Furthermore, we explore the decoding process, addressing the inherent uncertainty associated with quantum measurements and discussing strategies to minimize such uncertainty. Our findings suggest that hybrid encoding can improve quantum information processing tasks, making it a promising technique for future quantum computing applications. Further research is needed to optimize the encoding and decoding processes and explore the full potential of this approach in various quantum algorithms.**

Keywords— hybrid quantum encoding, amplitude encoding, basis encoding, quantum information processing, data storage, computational efficiency, quantum computing, quantum algorithms

I. Introduction

Quantum computing promises to revolutionize how we process and compute information by leveraging the unique properties of quantum mechanics. This technology aims to surpass classical computing in terms of speed and computational power, known as quantum supremacy[1]. Quantum computers can perform better than classical computers for specific computational problems, which require immense computational resources or time when solved using classical computational methods. Some significant problems where quantum computers can exhibit a significant advantage include Integer factorization, unstructured search, optimization problems, quantum simulation, and Machine learning. Quantum computers use quantum phenomena, such as superposition and entanglement, to achieve quantum supremacy, which allows them to perform complex calculations simultaneously and more efficiently than classical computers.

A significant challenge in leveraging the capabilities of quantum computing is how to adapt diverse classical problems to a quantum framework, which would enable the efficient use of quantum resources. This research concentrates on search problems involving large-scale data, emphasizing developing methods for converting data from classical systems to work effectively with quantum computers, thereby accelerating the processing time.

This research focuses on transforming and optimizing classical search problems for quantum computing to use the full potential of quantum technology for data-intensive applications by harnessing the inherent advantages of quantum computing for processing and analyzing large-scale data more efficiently than classical computers. The primary objective is to develop efficient data preparation techniques for converting large-scale unstructured data string, DNA data, into a format suitable for processing by quantum computers without considering noise or any physical implementation constraints. By creating specialized algorithms and data preparation methods for gate-based quantum computers, this research aims to demonstrate the potential for significant data preparation improvements in large-scale string search problems.

II. Quantum Computer and Pattern Search

A. Quantum Computer

There are two primary types of quantum computers based on their approach to computation: gate-based quantum computing and adiabatic quantum computing. Each method has its own advantages and challenges, offering unique ways to process and manipulate quantum information.

Gate-based quantum computing [2] is a series of quantum logic gates that perform computations on qubits. Quantum gates manipulate qubits' state through operations specifically designed to use the principles of quantum mechanics, such as superposition and entanglement. A well-known example of a gate-based quantum algorithm is Shor's algorithm,

Adiabatic quantum computing [3] is fundamentally different approach that relies on the principle of quantum annealing or adiabatic quantum optimization. This method gradually evolves the quantum system from an initial state to a final state by slowly changing the system's Hamiltonian. This mathematical construct represents the total energy of the system. The goal is to find the lowest energy state corresponding to the optimal solution to a given problem. Unlike gate-based quantum computing, adiabatic quantum

computing does not require quantum gates to manipulate qubits. Instead, it relies on a continuous transformation of the system's Hamiltonian, which, if performed slowly enough, the result will be optimal.

Both types of Quantum Computers leverage three significant quantum phenomena: quantum interference, quantum parallelism, and quantum tunneling.

Quantum interference [4] is a phenomenon that allows quantum computers to manipulate the probability amplitudes of qubits. It enables quantum algorithms to perform tasks more efficiently by discarding incorrect and reinforcing correct solutions.

Quantum parallelism [5] takes advantage of the superposition principle, allowing quantum computers to process multiple inputs simultaneously. It implies that a quantum bit, or qubit, can simultaneously represent both 0 and 1, in contrast to classical bits, which represent either 0 or 1 exclusively.

Quantum tunneling [6] is particularly relevant for adiabatic quantum computing, as it enables the system to overcome energy barriers and find the global minimum of the optimization problem more efficiently. This phenomenon allows the quantum system to explore different solutions simultaneously, escaping local minima and transitioning to more optimal configurations. By incorporating quantum tunneling, adiabatic quantum computers can solve complex optimization problems and combinatorial searches more efficiently than classical computing methods, making it a crucial component in the overall quantum computing landscape.

B. DNA Sequence Data

DNA sequences contain information about the genetic makeup of an organism, but the sequence itself does not have a defined format or organization. Analysis of DNA data often involves pattern recognition and statistical analysis to identify genetic markers or mutations.

Pixel data from images or videos are also considered unstructured data. Each pixel in an image or frame of a video contains color or brightness information, but the arrangement of the pixels does not have a defined structure. Image and video analysis involve techniques such as computer vision and machine learning to identify objects or patterns in the data. This research focuses specifically on string matching for string data. The same approach could be potentially applied to other types of unstructured data, including pixel data and video. However, further research and development would be needed to adapt the approach to these other data types.

C. Quantum Algorithms for Pattern Matching

Grover's algorithm [7] presents a complexity of O(√n), where n is the size of the unsorted database. This quadratic speedup significantly improves, especially when dealing with large databases. However, Grover's algorithm is specifically designed for searching unsorted databases. If the database is sorted, classical algorithms like binary search can achieve a complexity of O(log n) faster than Grover's algorithm in that specific scenario. For large-scale data, Grover's algorithm needs efficient data preparation and encoding. Also, preparing the quantum state representing the database and implementing the oracle function can be challenging, as it requires effective techniques to encode and manipulate the data within a quantum framework. For searching capabilities, Grover's algorithm is designed to locate a single occurrence of a target item within an unsorted database.

Grover's algorithm can be applied to accelerate pattern matching in genome sequencing, such as targeting complex RNA secondary structures. These structures, which are crucial for understanding the function and regulation of RNA, pose significant challenges for traditional pattern-matching algorithms due to their intricate folding patterns and interactions.

D. Encoding Classical Data into Qbit

Many quantum information processing applications require encoding classical data into quantum states.

Basis Encoding represents each bit of the classical data represented by an individual qubit, with the state $|0\rangle$ or $|1\rangle$ indicating the value. For example, the classical data 1010 is $|1\rangle|0\rangle|1\rangle|0\rangle$. Amplitude Encoding [8]represents classical data by the amplitudes of a quantum state. Angle Encoding represents classical data by the rotation angles of a set of gates applied to a single qubit. IQP [9]Encoding represents the classical data by the pattern of gates in a particular circuit architecture. Hamiltonian Evolution Ansatz [10]represents classical data by encoding it into a Hamiltonian, which evolved to prepare the desired quantum state.

III. Encoding Design

This section provides a detailed overview of our methodology for encoding DNA data. First, we provide detail of the characteristics of DNA data and how we have approached the problem of encoding it using quantum computing techniques. Next, our architecture comprises an interface of classical and quantum computers, allowing us to take advantage of the strengths. The Loop-Qbit encoder is the main objective of the research. Additionally, we will use a quantum computing simulator to test our approach before moving on to experimental implementations.

A. Amplitude Encoding

Given wave function.

$$|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i \ |i> |0>^{\otimes m} \tag{1}$$

where $N=2^m$ is the number of possible states for the m qubits, i is the state with index i and α is the amplitude corresponding to state *i*.

Amplitude Encoding is encoding data value into amplitude of the wave function ψ_A.

$$|\psi_A> = \sum_{i=0}^{N-1} \sqrt{\frac{x_i}{\sum_{j=0}^{N-1} x_j}} \ |i> |0>^{\otimes m} \tag{2}$$

where x is the value of the data element indexed by *i*

B. Hybrid Encoding

Hybrid encoding is a method of encoding data using both basis encoding and amplitude encoding. The value x_i of each data point is encoded into the amplitude of a qubit using amplitude encoding. The value *b* of each data point (0 or 1) is encoded into the basis of a qubit using basis encoding. The

resulting quantum state is a superposition of all the basis states, with the amplitude of each basis state determined by the corresponding value of *x*.

Given Data S = [x, b] where x is an integer in the range 0 to N-1 and b is either 0 or 1encoding steps are as follow:

1.Normalize data S: $x_{norm} = x/(N-1)$, $b_{norm} = b$

2.Apply amplitude encode to x from (2) we have

$$|\psi_x> = \sqrt{x_{norm}}|0> + \sqrt{1 - x_{norm}}\,|1> \quad (3)$$

3. Apply basis encoding to the b

$$|\psi_b> = |b> \quad (4)$$

4, Apply tensor product of two encoding states to get hybrid encoding

$$|\psi_{XB}> = |\psi_x> \otimes |\psi_b> \quad (5)$$

To decode the hybrid encoded state $|\psi_{XB}>$ A tensor product (5) of the encoded states $|\psi_x>$ and $|\psi_b>$, we can write the hybrid encoded as:

$$|\psi_{XB}> = (a\,|0> + b\,|1>) \otimes |\psi_b> \quad (6)$$

where $a = \sqrt{x_{norm}}$, $b = \sqrt{1 - x_{norm}}$

To decode b, we can measure the second register (the one containing $|\psi_b>$) on a computational basis. The outcome will be either state |0> or |1>, corresponding to the original value of b.

$State_b = 0$ when $|\psi_b> = |0>$

$State_b = 1$ when $|\psi_b> = |1>$

To decode *x*, apply an inverse amplitude encoding transformation, which can be realized using a controlled rotation gate. After applying the inverse amplitude encoding, we can measure the first register in the computational basis, which will give us an estimate of x_{norm}.

To reconstruct *x* and *b*

$x = \text{round}\,(x_{norm} * (N - 1))$

$b = State_b$

IV. Bit-Partition Encoding for DNA Data

Hybrid encoding combines multiple encoding methods, from classical data to qubit encoding. 1s and 0s to set the state of the basic encoding part in hybrid encoding, the data needs to be prepared by mapping each 1 and 0. We call this step Bit-Partition (BP).

A. DNA Data

In the context of DNA sequence data, this involves searching for specific patterns or sequences within extensive genetic information. In computational terms, DNA sequence data is represented as strings of four nucleotides: adenine (A), cytosine (C), guanine (G), and thymine (T). These are stored in specific file formats designed to manage genetic information. In this research, we use DNA data stored in FASTA format [11]. The FASTA format is a widely used text-based format for representing DNA sequences as shown in Figure 1. It consists of a single-line description, followed by the sequence, where each character represents a nucleotide. The description line starts with a '>' symbol, followed by a unique identifier and additional metadata. The simplicity and human-readability of the FASTA format make it a popular choice for storing and sharing genetic data.

```
>Sequence1 DNA sequence from organism X
ATCGATCGATCGATCGATCGATCGATCGATCG
ATCG
```

Fig. 1. Example of a FASTA file

B. Bit-Partition Hybrid Encoding

A classical preprocessing step is used to convert a FASTA DNA sequence into a format suitable for performing the encoding. Using FASTA DNA as the test data, the algorithm inputs a DNA sequence of length n and outputs the bitstream partitions for each nucleotide (A, G, C, T). During the classical preprocessing, it initializes four lists to all zeros, representing the partitions for each nucleotide. Then, for each character in the DNA sequence, it sets the corresponding entry in the appropriate partition list to 1 and sets the entries in the other three partition lists to 0. Finally, it returns the four partition lists, which can be further used for encoding. The Pseudo Code is shown in Figure 2.

```
input: DNA sequence S of length n
output: Bitstream partitions for each nucleotide, partition_A,
partition_G, partition_C, partition_T

initialize four lists of length n, partition_A, partition_G, partition_C,
and partition_T, to all zeros

for i from 1 to n do
   if S[i] is 'A' then
      partition_A[i] ← 1
      partition_G[i] ← 0
      partition_C[i] ← 0
      partition_T[i] ← 0
   else if S[i] is 'G' then
      partition_A[i] ← 0
      partition_G[i] ← 1
      partition_C[i] ← 0
      partition_T[i] ← 0
   else if S[i] is 'C' then
      partition_A[i] ← 0
      partition_G[i] ← 0
      partition_C[i] ← 1
      partition_T[i] ← 0
   else if S[i] is 'T' then
      partition_A[i] ← 0
      partition_G[i] ← 0
      partition_C[i] ← 0
      partition_T[i] ← 1
return partition_A, partition_G, partition_C, partition_T
```

Fig. 2. Pseudo Code of Bit-Partition Hybrid Encoding

C. Architecture

The BP hybrid encoding is applied to DNA data by implementing four separate systems, each corresponding to

one of the four nucleotides (adenine, guanine, cytosine, and thymine). The algorithm can be summarized as follows:

1. Initialize four quantum registers set for each nucleotide (A, G, C, and T), each consisting of 8 qubits (2 qubits per nucleotide).
2. Prepare the input data by converting the DNA sequence into a bitstream representation. For each character in the DNA sequence, set the corresponding entry in the appropriate partition list to 1 and the entries in the other three partition lists to 0.
3. Encode the input data into the quantum registers using the hybrid encoding scheme for each nucleotide, encode the index number into the amplitude of the qubits, and set the qubit state according to the value in the second column of the matrix (either 0 or 1)
4. Execute quantum operations on the encoded data as the specific application requires.
5. Measure the resulting wave function to obtain the desired probability distribution over the data elements.

D. Result

The BP hybrid encoding scheme uses basis and amplitude encoding for unstructured data, such as DNA data. The data to be encoded is an *i* x 2 matrix where the first column, *i*, represents the index, and the second column represents the value. The objective is to encode the index number into the amplitude of the qubit while setting the state of the qubit according to the value in the second column of the matrix. In this research, we use a 64x2 matrix; the encoding uses 2 qubits per nucleotide. The resulting wave function can be measured to obtain the desired probability distribution over the data elements. An example of encoding and decoding of nucleotide A is shown in Figure 3, and the result analysis is shown in Figure 4.

Given A = [0110…10]
SA = [[0, 0], [1, 1], [2, 1], [3, 0]…[62,1],[63,0]]

Output State vector
Data point [0, 0]: State vector = ['0.0+0.0j', '0.0+0.0j', '1.0+0.0j', '0.0+0.0j']
Data point [1, 1]: State vector = ['0.0+0.0j', '0.12598816+0.0j', '0.0+0.0j', '0.99203175+0.0j']
Data point [2, 1]: State vector = ['0.0+0.0j', '0.17817416+0.0j', '0.0+0.0j', '0.98399897+0.0j']
Data point [3, 0]: State vector = ['0.21821789+0.0j', '0.0+0.0j', '0.97590007+0.0j', '0.0+0.0j']
Data point [62, 1]: State vector = ['0.0+0.0j', '0.99203175+0.0j', '0.0+0.0j', '0.12598816+0.0j']
Data point [63, 0]: State vector = ['1.0+0.0j', '0.0+0.0j', '0.0+0.0j', '0.0+0.0j']

Decode and Reconstructed
SA = [[0, 0], [1, 1], [2, 1], [3, 0]…[62,1],[63,0]]

Fig. 3. Encoder and Decoder result

The results analysis in Figure 4 demonstrated a hybrid encoding technique combining amplitude and basis encoding. We represented the index of the data points using amplitude encoding, while basis encoding represented the actual data. Specifically, we encoded the given dataset's first column (indices) with amplitude encoding and the second column (data) with basis encoding. This approach allowed us to represent and analyze classical data, such as DNA sequences with AGCT converted to 1 or 0 on its bit string, using quantum computing concepts and tools like Python and Qiskit.

Normalization factors: N = 64
Data point [0, 0]: Normalized value: 0/63
Data point [1, 1]: Normalized value: 1/63
Data point [62, 1]: Normalized value: 62/63
Data point [63, 0]: Normalized value: 63/63

Data point [0, 0]
State vector: [0.+0.j, 0.+0.j, 1.+0.j, 0.+0.j]
Decode the first qubit amplitude: |0|^2 + |0|^2 = 0
Basis states and amplitudes: |00⟩: 0, |01⟩: 0, |10⟩: 1, |11⟩: 0
The highest amplitude is the basis state |10⟩; the second qubit decode is 0.
Reconstruct x_0 = 0/63 * 63 = 0
Reconstruct b_0 = 0

Data point [1, 1]
State vector: [0.+0.j, 0.12598816+0.j, 0.+0.j, 0.99203175+0.j]
Decode the first qubit amplitude: |0|^2 + |0.12598816|^2 ≈ 1/63
Basis states and amplitudes: |00⟩: 0, |01⟩: 0.12598816, |10⟩: 0, |11⟩: 0.99203175
The highest amplitude is the basis state |11⟩; the second qubit decode is 1.
Reconstruct x_1 = 1/63 * 63 = 1
Reconstruct b_1 = 1

Data point [2, 1]
State vector: [0.+0.j, 0.17817416+0.j, 0.+0.j, 0.98399897+0.j]
Decode the first qubit amplitude: |0|^2 + |0.17817416|^2 ≈ 2/63
Basis states and amplitudes: |00⟩: 0, |01⟩: 0.17817416, |10⟩: 0, |11⟩: 0.98399897
The highest amplitude is the basis state |11⟩; the second qubit decode is 1.
Reconstruct x_2 = 2/63 * 63 = 2
Reconstruct b_2 = 1

Data point [3, 0]
State vector: [0.21821789+0.j, 0.+0.j, 0.97590007+0.j, 0.+0.j]
Decode the first qubit amplitude: |0.21821789|^2 + |0|^2 ≈ 3/63
Basis states and amplitudes: |00⟩: 0.21821789, |01⟩: 0, |10⟩: 0.97590007, |11⟩: 0
The highest amplitude is the basis state |10⟩; the second qubit decode is 0.
Reconstruct x_3 = 3/63 * 63 = 3
Reconstruct b_3 = 0

Data point [62, 1]
State vector: [0.+0.j, 0.99203175+0.j, 0.+0.j, 0.12598816+0.j]
Decode the first qubit amplitude: |0|^2 + |0.99203175|^2 ≈ 62/63
Basis states and amplitudes: |00⟩: 0, |01⟩: 0.99203175, |10⟩: 0, |11⟩: 0.12598816
The highest amplitude is the basis state |01⟩; the second qubit decode is 1.
Reconstruct x_{62} = 62/63 * 63 = 62
Reconstruct b_{62} = 1

Data point [63, 0]
State vector: [1.+0.j, 0.+0.j, 0.+0.j, 0.+0.j]
Decode the first qubit amplitude: |1|^2 + |0|^2 = 1
Basis states and amplitudes: |00⟩: 1, |01⟩: 0, |10⟩: 0, |11⟩: 0
The highest amplitude is the basis state |00⟩; the second qubit decode is 0.
Reconstruct x_{63} = 1 * 63 = 63
Reconstruct b_{63} = 0

SA = [[0, 0], [1, 1], [2, 1], [3, 0]…[62,1], [63,0]]

Fig. 4. Result Analysis

The encoding process involved normalizing the index values in the first column and encoding these normalized values into the amplitudes of a quantum state. We then used basis encoding to represent the second column of the dataset in the binary representation of qubits. As a result, we could represent the dataset in the form of state vectors and basis states, which are fundamental elements in quantum

computing. With this technique, we can maintain the quantum phenomena leveraged by quantum computers: quantum interference, quantum parallelism, and quantum tunneling.

Hybrid encoding combines amplitude encoding for representing the index of data points and basis encoding for representing the actual data. It balances each method's strengths and weaknesses. Amplitude encoding can efficiently represent continuous data but may suffer from issues like poor resolution and susceptibility to noise. On the other hand, basis encoding offers a more robust representation of discrete data but requires more qubits for higher-dimensional data. By using amplitude encoding for the index of data points and basis encoding for the actual data, we can take advantage of the efficiency of amplitude encoding for continuous index values while leveraging the robustness of basis encoding for discrete data. This combination mitigates some drawbacks of using either encoding method alone, providing a more versatile and practical approach to representing classical data in a quantum system.

E. Conclusion

The advantage of using the implemented BP hybrid encoding for strings is that it allows us to represent each character in the string as a separate array, which can be used for certain operations such as sorting and searching. For example, BP hybrid encoding can make this task more efficient if we have a large dataset of strings and we want to search for all the occurrences of a specific character within those strings. This encoding can also help reduce string data storage requirements. In some cases, strings may contain repeated patterns of characters or words. BP hybrid encoding can identify and store these patterns as separate arrays, which can then be reused across multiple strings, leading to significant savings in memory. This encoding method can be integrated with quantum algorithms, so we can harness the full potential of quantum technology and improve the performance of various operations on large-scale data.

Overall, BP hybrid encoding is a flexible and efficient way to represent string data, which can help improve the performance of various operations on that data while reducing storage requirements.

F. Future work

Our future work aims to extend the BP hybrid encoding method's capabilities to handle better various types of data, including pixel data with color depth values. Its improvements will make the hybrid encoding method more versatile and applicable to various applications. An example is pixel data which may use a color quantization technique to reduce the number of colors used in the image.

BP Hybrid encoding can be implemented with both gate-based and annealing-based quantum computers. Gate-based quantum computers offer high flexibility and can implement arbitrary unitary operations, making them suitable for many quantum algorithms. On the other hand, annealing-based quantum computers have a simpler architecture and are better suited for optimization problems.

REFERENCES

[1] C. Palacios-Berraquero, L. Mueck, and D. M. Persaud, “Instead of ‘supremacy’ use ‘quantum advantage,’” *Nature*, vol. 576, no. 7786, pp. 213–213, December 2019. doi:10.1038/d41586-019-03781-0

[2] K. Michielsen, M. Nocon, D. Willsch, F. Jin, T. Lippert, and H. De Raedt, “Benchmarking gate-based quantum computers,” *Computer Physics Communications*, vol. 220, pp. 44–55, November 2017. doi:10.1016/j.cpc.2017.06.011

[3] C. C. McGeoch, R. Harris, S. P. Reinhardt, and P. I. Bunyk, “Practical Annealing-Based Quantum Computing,” *Computer*, vol. 52, no. 6, pp. 38–46, June 2019. doi:10.1109/MC.2019.2908836

[4] R. C. Liu, B. Odom, Y. Yamamoto, and S. Tarucha, “Quantum interference in electron collision,” *Nature*, vol. 391, no. 6664, pp. 263–265, January 1998. doi:10.1038/34611

[5] P. W. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, October 1997. doi:10.1137/S0097539795293172

[6] M. Razavy, *Quantum theory of tunneling*. River Edge, NJ: World Scientific, 2003.

[7] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, Philadelphia, Pennsylvania, United States: ACM Press, 1996, pp. 212–219. doi:10.1145/237814.237866

[8] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, 10th anniversary ed. Cambridge ; New York: Cambridge University Press, 2010.

[9] D. Beaulieu, D. Miracle, A. Pham, and W. Scherr, “Quantum Kernel for Image Classification of Real World Manufacturing Defects.” arXiv, December 2022. [Online] http://arxiv.org/abs/2212.08693.

[10] S. Stanisic *et al.*, “Observing ground-state properties of the Fermi-Hubbard model using a scalable algorithm on a quantum computer,” *Nature Communications*, vol. 13, no. 1, p. 5743, October 2022. doi:10.1038/s41467-022-33335-4

[11] D. Pratas, M. Hosseini, and A. J. Pinho, “Cryfa: A Tool to Compact and Encrypt FASTA Files,” in *11th International Conference on Practical Applications of Computational Biology & Bioinformatics*, F. Fdez-Riverola, M. S. Mohamad, M. Rocha, J. F. De Paz, and T. Pinto, Eds., Cham: Springer International Publishing, 2017, pp. 305–312.