Research on Optimization Algorithms for Integer Decomposition Approximation of DFT Matrix with Sparsity Constraints

Zhongchuan Zhou *

School of Mathematical Sciences, Chongqing Normal University, Chongqing, China, 401331

* Corresponding Author Email: 2022110510102@stu.cqnu.edu.cn

Abstract. The Discrete Fourier Transform (DFT), as an important mathematical tool in the field of communications, has been widely applied in signal processing and channel estimation. However, with the continuous growth in antenna arrays, the number of channels, and communication bandwidth, the computational resources required to implement the FFT on a chip have been increasing significantly. Therefore, there is a need for more efficient and simplified methods to implement the DFT, aiming to reduce computational complexity and resource consumption without compromising accuracy. To address this issue, this paper adopts an approximation approach for the DFT, which approximates the DFT matrix as a product of a series of sparse matrices with limited element values. Based on this approach, this paper proposes an integer matrix decomposition approximation algorithm with sparsity constraints, aimed at reducing the computational complexity of the DFT and optimizing the computational process. Numerical experiments demonstrate that, compared to the traditional FFT algorithm, the proposed method reduces computational complexity by approximately fivefold, highlighting its significant advantages and practical engineering value.

Keywords: Approximate DFT, Integer Decomposition Approximation, Optimization Algorithm.

1. Introduction

The Discrete Fourier Transform (DFT) is a fundamental linear transformation widely utilized in engineering, physics, and mathematics. In communication signal processing, the DFT is crucial for time-frequency domain transformations in Orthogonal Frequency Division Multiplexing (OFDM) systems [1-2]. Moreover, DFT-based pre-beamforming matrices are nearly optimal for Uniform Linear Arrays (ULAs) and require only coarse information regarding angle and angular spread [3].

In practical engineering applications, the hardware complexity of DFT computation is influenced by both algorithmic complexity and the range of data element values. Greater algorithmic complexity and larger data value ranges lead to increased hardware complexity. The Fast Fourier Transform (FFT) algorithm is commonly employed to efficiently implement the DFT by leveraging various properties, significantly reducing computational complexity [4]. However, as wireless communication technologies evolve, larger antenna arrays, an increasing number of channels, and expanding communication bandwidth heighten the demand for FFT, resulting in greater hardware overhead for implementation on specialized chips. Thus, developing more efficient and simplified methods for DFT implementation is essential to reduce computational complexity and facilitate practical engineering applications.

The quest for efficient DFT implementations that minimize multiplication complexity and enhance computational efficiency is a multidisciplinary research area encompassing signal processing, computer science, and applied mathematics, with various algorithms available [5]. The FFT algorithm remains the most prevalent for implementing the DFT. Three main categories of popular FFT algorithms exist: Cooley-Tukey FFT [6], Split-Radix FFT [7], and Winograd FFT [8].

The Cooley-Tukey FFT is a classical approach that reduces computational load by recursively decomposing the DFT into smaller DFTs. The Split-Radix FFT optimizes the FFT process by employing different radices (e.g., 2, 3, 5) at different stages, further decreasing the number of multiplications. The Winograd FFT, optimized for small-size DFTs (e.g., 8-point or 16-point), utilizes

specific matrix decomposition techniques to reduce multiplication operations, making it particularly suitable for hardware implementation.

R. E. Blahut et al. [9] established the theoretical lower bound of the multiplication complexity for the DFT as a function of N, indicating that the computational complexity for performing an N-point DFT directly is $O(N^2)$. All FFT algorithms can provide an exact implementation of the DFT by using the sparse decomposition of the DFT matrix, with a computational complexity approaching this lower bound. However, such high-precision implementations have significant limitations in practical applications. For example, in the field of communications, as wireless communication technology evolves, antenna arrays are growing larger, the number of channels is increasing, and communication bandwidth is expanding. This results in a growing demand for FFT, which in turn leads to increased hardware overhead when implementing FFT on specialized chips. The computational complexity of traditional FFT algorithms grows quadratically with the size of the DFT matrix, making this complexity impractical for most engineering applications. High-precision FFT implementations can lead to higher power consumption, increased latency, and a greater likelihood of system failures.

To address the aforementioned issues, the use of Approximate DFT (ADFT) algorithms opens up new possibilities for fast computation. Unlike the high-precision implementation of traditional FFT algorithms, the core idea of ADFT is to approximate the DFT matrix as a product of a series of sparse matrices with limited element values [10]. This matrix decomposition approach allows for the efficient handling of large-scale datasets while maintaining low computational complexity. Although ADFT does not compute the DFT in a mathematically exact sense, it is sufficiently accurate because the ADFT algorithm achieves much greater efficiency compared to the theoretical lower bound of the multiplication complexity for the N-point DFT as presented in [11]. ADFT is essentially a trade-off between accuracy and complexity, sacrificing some computational precision to significantly reduce complexity [12].

This paper, starting from the properties of the DFT matrix, proposes an integer matrix decomposition approximation algorithm with sparsity constraints based on the core idea of ADFT. The goal is to reduce the computational complexity of DFT and optimize the computation process.

2. Theory and Methods

2.1. Integer Matrix Decomposition Approximation Algorithm with Sparsity Constraints

To understand the proposed ADFT algorithm, this paper first discuss the definition of the DFT and the mathematical background related to the FFT algorithm.

Given an N-point one-dimensional complex signal in the time domain $x_0,\ x_1,\ \ldots x_{N-1}$, the discrete Fourier transform (DFT) produces a complex signal $X_k(k = 0,1,\ldots,N-1)$as given by:

$$X_k = \sum_{n=0}^{N-1}\left(x_n e^{\frac{-i2\pi nk}{N}}\right), k = 0,1,\ldots,N-1 \tag{1}$$

In matrix form, this can be expressed as:

$$X = F_N x \tag{2}$$

Where $\mathbf{x} = \left[x_0, x_1, \ldots, x_{N-1}\right]^\top$ is the time-domain signal vector, $\mathbf{X} = \left[\mathrm{X}_0, \mathrm{X}_1, \ldots, \mathrm{X}_{N-1}\right]^\top$ is the frequency-domain signal vector, and F_N is the DFT matrix, given by:

$$\mathbf{F}_N = \frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & w & \cdots & w^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & w^{N-1} & \cdots & w^{(N-1)(N-1)} \end{bmatrix}, \quad w = \mathrm{e}^{-\frac{i2\pi}{N}} \tag{3}$$

In practical engineering applications, this matrix form not only systematizes and clarifies the computation of the DFT but also provides a foundation for optimizing the DFT computation by utilizing various matrix operations and properties from linear algebra.

The DFT can be accurately implemented using traditional FFT methods; however, the complexity of parallel FFT implementations is relatively high, especially for large-scale data or applications requiring real-time processing. Efficiently implementing FFT on multi-core or distributed systems remains a challenge [13]. Therefore, when output precision requirements are less stringent, an alternative approach can be considered: utilizing a matrix product fitting method. The core idea of this approach is to approximate the DFT matrix as a product of a series of sparse matrices with limited element values. For example, the DFT matrix F_N can be approximated as $\mathbf{F}_N \approx \mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K$,where each $A_i, i=1,\ldots,K$ is a sparse matrix with integer elements.

Considering that the hardware complexity of DFT implementation primarily arises from multiplication operations, this paper proposes an integer matrix decomposition approximation algorithm with sparsity constraints to reduce the number of multiplications and optimize the computation process.

For a given N-dimensional DFT matrix F_N, This study designs K matrices $\mathcal{A}=\{\mathbf{A}_1,\mathbf{A}_2,\ldots,\mathbf{A}_K\}$ such that the matrix F_N is approximated as closely as possible to $\beta\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K$ in terms of the Frobenius norm, i.e.,

$$\min_{\mathcal{A},\beta} \mathrm{RMSE}(\mathcal{A},\beta)=\frac{1}{N}\sqrt{\left\|\mathbf{F}_N-\beta\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\right\|_F^2} \tag{4}$$

where β is a real-valued scaling factor. Compared to multiplication, addition has much lower hardware complexity. Therefore, the focus of this research is on the hardware complexity of multipliers: $C=q\times L$, where C denotes the hardware complexity, q represents the number of multiplications per operation, and L denotes the number of operations.

In this context, q denotes the range of values in the decomposed matrices A_k . In this work, the real and imaginary parts of the elements in A_k are limited to the set $\mathcal{P}=\left\{0,\pm1,\pm2,\ldots,\pm2^{q-1}\right\}$. For example, if $\mathcal{P}=\{0,\pm1,\pm2,\pm4\}$, then q=3. L represents the number of complex multiplications, where multiplications by 0 , ±1 , $\pm j$ are not counted as complex multiplications. For instance, if $\mathcal{P}=\{0,\pm1,\pm2,\pm4\}$, then the computational complexity C=6 (where q=3 and L=2) for the following matrix multiplication is: $\begin{bmatrix}1 & 2+4j\\ 1+2j & 0\end{bmatrix}\times\begin{bmatrix}0 & 1\\ 2+4j & 2-4j\end{bmatrix}$. Consider the following constraints:

Constraint 1: [**Sparsity Constraint]** Limit each matrix A_k to have at most 2 non-zero elements per row.

To facilitate problem formulation, a new variable is introduced $M_k(i,j)$, defined as:

$$M_k(i,j)=\begin{cases}1, A_k(i,j)\neq 0\\ 0, A_k(i,j)=0\end{cases} \tag{5}$$

Thus, Constraint 1 can be expressed as:

$$\sum_{j=1}^{N} M_k(i,j)\leqslant 2, k=1,2,\ldots K, i=1,2,\ldots N \tag{6}$$

Constraint 2: [**Feasibility Domain Constraint]** Limit the value range of each element in every matrix A_k such that:

$$\mathbf{A}_k[l,m]\in\{x+jy\mid x,y\in\mathcal{P}\} \tag{7}$$

where $\mathcal{P}=\left\{0,\pm1,\pm2,\ldots,\pm2^{q-1}\right\}, k=1,2,\ldots,K; l,m=1,2,\ldots,N$ and $\mathbf{A}_k[l,m]$ represents the element in the l-th row and m-th column of matrix A_k.

As can be seen, the matrix elements $\mathbf{A}_k[l,m]$ are all integers, which reduces the complexity of each multiplier; additionally, the sparsity of each A_k reduces the number of multiplication operations. Under the constraints of Constraint 1 and Constraint 2, the original DFT matrix F_N can be approximately fitted by matrix multiplication, i.e., $\mathbf{F}_N \approx \mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K$. This approach is essentially a trade-off between precision and hardware complexity, where some computational precision is sacrificed in exchange for significantly reduced hardware complexity.

Based on the above analysis, under the constraints of both sparsity and value range for A_k, the following mathematical optimization model can be established:

$$\min_{\mathcal{A},\beta} \mathrm{RMSE}(\mathcal{A},\beta)=\frac{1}{N}\sqrt{\left\|\mathbf{F}_N-\beta\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\right\|_F^2} \tag{8}$$

Subject to:

$$\begin{cases}\sum_{j=1}^{N} M_k(i,j)\leqslant 2, \quad k=1,2,\ldots,K, \quad i=1,2,\ldots,N\\ \mathcal{P}_{\min}\leqslant \mathrm{Re}\left(\mathcal{A}_k(i,j)\right)\leqslant \mathcal{P}_{\max}\\ \mathcal{P}_{\min}\leqslant I_m\left(\mathcal{A}_k(i,j)\right)\leqslant \mathcal{P}_{\max}\\ \beta\in\mathbb{R}, \quad \mathcal{P}=\{0,\pm1,\pm2,\pm4,\pm5\}\end{cases} \tag{9}$$

The objective function $\mathrm{RMSE}(\mathcal{A},\beta)$ aims to minimize the Root Mean Square Error between the original DFT matrix F_N and the approximated matrix $\beta\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K$.

The constraints ensure that each matrix A_k is sparse with at most 2 non-zero elements per row and that the real and imaginary parts of each element in A_k fall within the specified range $[\mathcal{P}_{\min},\mathcal{P}_{\max}]$.β is a real-valued scaling factor, and $\mathcal{P}$ denotes the set of permissible integer values for the matrix elements.

This optimization model balances the trade-off between approximation accuracy and computational efficiency by enforcing sparsity and limiting the value range of matrix elements. The mathematical model described above is essentially a non-convex and non-continuous constrained optimization problem. To address this problem, this paper employs an "alternating optimization and selection" genetic algorithm [14] to determine the parameter β and the decomposition matrices. The genetic algorithm is a heuristic search algorithm based on simulating biological evolution and Darwinian principles. Initially, a feasible set of solutions is specified. Then, through genetic operations such as selection, crossover, and mutation, the algorithm finds the optimal solution under the given constraints. The design of the algorithm is as follows:

Algorithm 1 Matrix Decomposition Based on Genetic Algorithm

Require: Number of matrices K, population size N, DFT matrix F_N, maximum iterations M

Ensure: Optimal parameters β, $A_1, A_2, \ldots, A_K$

1. Initialize: N=0
2. $Q = \{\beta, A_1, A_2, \ldots, A_K\} \leftarrow N$
3. **While** {N < M} **do**
4. $\min_{\mathcal{A},\beta} \mathrm{RMSE}(\mathcal{A},\beta) = \frac{1}{N}\sqrt{\left\|\mathbf{F}_N - \beta\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\right\|_F^2}$
5. **If** $f(Q^i) \le f(Q^*)$ **then**
6. $Q^* \leftarrow Q^i$
7. $f^* \leftarrow f(Q^i)$
8. **End if**
9. $i \leftarrow i+1$
10. **End while**
11. **Return** $f(Q^*), Q^* = \{\beta^*, A_1^*, A_2^*, \ldots, A_K^*\}$
12.

2.2. Large-Scale DFT Matrix Decomposition Problem Based on Divide-and-Conquer Strategy

Building on the model and algorithm proposed in the previous section, this work adopts matrix decomposition under sparsity constraints and integrates a divide-and-conquer strategy to address larger-scale DFT matrix decomposition problems. An in-depth analysis of the computational complexity is also provided. For clarity, the definition of the Kronecker product of matrices is first introduced [15]: Let $X \in C^{m\times n}, Y \in C^{p\times q}, \otimes$ denote the Kronecker product of two matrices, which is defined as follows:

$$\mathbf{X}\otimes\mathbf{Y} = \begin{pmatrix} x_{11}\mathbf{Y} & x_{12}\mathbf{Y} & \cdots & x_{1n}\mathbf{Y} \\ x_{21}\mathbf{Y} & x_{22}\mathbf{Y} & \cdots & x_{2n}\mathbf{Y} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1}\mathbf{Y} & x_{m2}\mathbf{Y} & \cdots & x_{mn}\mathbf{Y} \end{pmatrix} \in \mathbb{C}^{(mp)\times(nq)} \tag{10}$$

After introducing the concept of the Kronecker product, the properties of the Kronecker product of matrices are presented [16]:

$$\begin{aligned} &1.\mathbf{X}\otimes\mathbf{Y}\otimes\mathbf{Z} = (\mathbf{X}\otimes\mathbf{Y})\otimes\mathbf{Z} = \mathbf{X}\otimes(\mathbf{Y}\otimes\mathbf{Z}) \\ &2.(\mathbf{X}+\mathbf{Y})\otimes\mathbf{Z} = \mathbf{X}\otimes\mathbf{Z}+\mathbf{Y}\otimes\mathbf{Z} \\ &3.(\mathbf{X}\otimes\mathbf{Y})(\mathbf{U}\otimes\mathbf{V}) = (\mathbf{XU})\otimes(\mathbf{YV}) \end{aligned} \tag{11}$$

Here, $\mathbf{X}\in\mathbb{C}^{m\times n}$, $\mathbf{Y}\in\mathbb{C}^{s\times t}$, $\mathbf{U}\in\mathbb{C}^{n\times p}$. In light of the preceding discussion and the divide-and-conquer strategy, an initial decomposition of a large-scale DFT matrix using the Kronecker product is considered. For example, decomposing the DFT matrices of size 4 and 8 results in the matrix $F_N = F_{N_1}\otimes F_{N_2}$. It is evident that F_N becomes a 32×32 matrix. Therefore, the decomposition problem of large-scale DFT matrices can be handled in three steps. Initially, a divide-and-conquer strategy based on the Kronecker product is adopted to decompose the large-scale DFT matrix F_N. After decomposing F_N, the smaller-sized DFT matrices F_{N_1}, F_{N_2} are further decomposed, incorporating the previously proposed algorithm to perform matrix decomposition. This process

yields a series of sparse matrices that satisfy the given constraints. Subsequently, the properties of the Kronecker product are used to combine the decomposed matrices. Finally, F_{N_1}, F_{N_2} is decomposed into a series of sparse matrix products, with calculations performed for the RMSE error and complexity C. The computational flow of this process is described as follows:

Step 1: Based on the Kronecker product, decompose the large-scale DFT matrix F_N, where $F_N = F_{N_1} \otimes F_{N_2}$. Utilizing the integer matrix decomposition approximation algorithm, based on the sparse constraints presented earlier, Further decomposition of F_{N1} and F_{N2} into sparse forms is achieved as follows:

$$F_{N_1} = F_{N_1}^{(1)} F_{N_1}^{(2)} \ldots F_{N_1}^{(K_1)}, \quad F_{N_2} = F_{N_2}^{(1)} F_{N_2}^{(2)} \ldots F_{N_2}^{(K_2)} \tag{12}$$

Thus, we approximately obtain:

$$F_N = F_{N_1} \otimes F_{N_2} \approx \left(F_{N_1}^{(1)}, F_{N_1}^{(2)}, \ldots, F_{N_1}^{(K_1)}\right) \otimes \left(F_{N_2}^{(1)}, F_{N_2}^{(2)}, \ldots, F_{N_2}^{(K_2)}\right) \tag{13}$$

Step 2: Utilizing the property of the Kronecker product , we obtain:

$$F_N \approx \left(F_{N_1}^{(1)} \otimes F_{N_2}^{(1)}\right)\left(F_{N_1}^{(2)} \otimes F_{N_2}^{(2)}\right) \ldots \left(F_{N_1}^{(K_1)} \otimes F_{N_2}^{(K_2)}\right) \tag{14}$$

Step 3: Then define:

$$A_1 = F_{N_1}^{(1)} \otimes F_{N_2}^{(1)}, \quad A_2 = F_{N_1}^{(1)} \otimes F_{N_2}^{(2)}, \ldots, \quad A_K = F_{N_1}^{(K_1)} \otimes F_{N_2}^{(K_2)} \tag{15}$$

Finally, calculate:

$$\min_{\mathcal{A},\beta} \mathrm{RMSE}(\mathcal{A}, \beta) = \frac{1}{N} \sqrt{\left\| \mathbf{F}_N - \beta \mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_K \right\|_F^2} \tag{16}$$

In the above process, although **Algorithm 1** initially satisfies the two constraint conditions for the decompositions of F_{N1} and F_{N2}, during the matrix combination in Step 2, the preservation of the Kronecker product properties cannot be guaranteed.

For each $A_i = F_{N_1}^{(K_1)} \otimes F_{N_2}^{(K_2)}, i = 1, 2, \cdots, K$, the matrix sparsity makes it impossible to ensure that all values lie within the feasible domain, indicating that the direct combination of these matrices as the final decomposition into $A_1, A_2, \ldots, A_K$ is not possible.

Therefore, this paper proposes leveraging the properties of the Kronecker product for combination after decomposing that after decomposing F_{N1} and F_{N2}. For each combination $\left(F_{N_1}^{(i)} \otimes F_{N_2}^{(i)}\right)$, verification is conducted to ensure that it satisfies constraint conditions 1 and 2. If these conditions are not met, F_{N1} and F_{N2} are re-decomposed until both constraints are satisfied simultaneously.The resulting computational flow is as follows:

Algorithm 2 Matrix Decomposition under Kronecker Product Constraints

Require: Number of matrices K_1, K_2

Ensure: $F_{N_1}^{(1)}, F_{N_1}^{(2)}, \ldots, F_{N_1}^{(K_1)}, F_{N_2}^{(1)}, F_{N_2}^{(2)}, \ldots, F_{N_2}^{(K_2)}$, $RMSE$, hardness C

Initialize

While i **do**

$F_{N_1} \approx F_{N_1}^{(1)} \cdot F_{N_1}^{(2)} \cdots F_{N_1}^{(K_1)}$

$F_{N_2} \approx F_{N_2}^{(1)} \cdot F_{N_2}^{(2)} \cdots F_{N_2}^{(K_2)}$

$A_i = F_{N_1}^{(i)} \otimes F_{N_2}^{(i)}$

If A_i satisfies Constraint 1 and Constraint 2 **then**

Return

Else

$i \leftarrow i+1$

End if

End while

Return $A_1, A_2, \ldots, A_K, \beta$

Based on Algorithm 2, a solution that simultaneously satisfies constraint conditions 1 and 2 for $A_1, A_2, \ldots, A_K$ can ultimately be obtained. From this point, the sparse structure can be followed, proceeding to matrix decompositions under constraint conditions using an improved evolutionary algorithm to achieve structured matrix decomposition.

Therefore, similarly, for large-scale DFT matrices, the following mathematical optimization model can be established:

$$\min_{\mathcal{A},\beta} \mathrm{RMSE}(\mathcal{A},\beta) = \frac{1}{N}\sqrt{\left\|\mathbf{F}_N - \beta \mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_K\right\|_F^2} \tag{17}$$

Subject to:

$$\begin{cases} \sum_{j=1}^{N} M_k(i,j) \leqslant 2, \quad k = 1,2,\ldots,K, i = 1,2,\ldots,N \\ \mathcal{P}_{\min} \leqslant \mathrm{Re}\left(\mathcal{A}_k(i,j)\right) \leqslant \mathcal{P}_{\max} \\ \mathcal{P}_{\min} \leqslant I_m\left(\mathcal{A}_k(i,j)\right) \leqslant \mathcal{P}_{\max} \\ \beta \in R, \mathcal{P} = \left\{0, \pm 1, \pm 2, \ldots, \pm 2^{q-1}\right\} \end{cases} \tag{18}$$

It can be seen that the above mathematical model is essentially a non-convex discrete optimization problem. Therefore, heuristic algorithms (or improved evolutionary algorithms) are employed to solve it. The detailed solution results and analysis will be provided in the following sections.

3. Numerical Analysis

3.1. Data Generation and Experimental Setup

First, based on the steps of Algorithm 1 proposed earlier in this paper, this paper use Python programming to perform numerical experiments, focusing on RMSE error analysis and hardware complexity analysis, and compare with the traditional FFT algorithm. The DFT matrix F_N is first considered for $N = 2^t, t = 1,2,3,4,5$, and optimize for $\mathcal{A}$ and β while satisfying constraint conditions

1 and 2. Through Python programming and the improved evolutionary algorithm, this paper perform numerical calculations based on the optimization model and calculate the minimum error and the hardware complexity C.

In addition, based on the basic steps of Algorithm 1 and Algorithm 2 proposed earlier, Python programming is also used to perform numerical experiments for l for large-scale DFT matrices F_N (taking N=1024 as an example) and compare the experimental results with the traditional FFT algorithm.

3.2. Analysis of experimental results

Using the improved evolutionary algorithm, the parameter values in the model were obtained, along with the optimized solution for t values. This paper calculated the RMSE errors and the hardware complexity C under the corresponding solutions, and the results are shown in **Table 1**.

Table 1. Numerical Calculation Results for Different Values of t

t	1	2	3	4	5
RMSE	0.0627	0.0957	0.2987	0.3403	0.3519
C	27	114	348	1256	1367
β	0.0084	0.0032	0.0025	0.0053	0.0018

Based on the numerical experiment results in **Table 1**, this paper found that the computational complexity C increases significantly with the increase in t. Next, this paper further analyze the RMSE error and computational complexity C, and obtain the following visualized graphs:

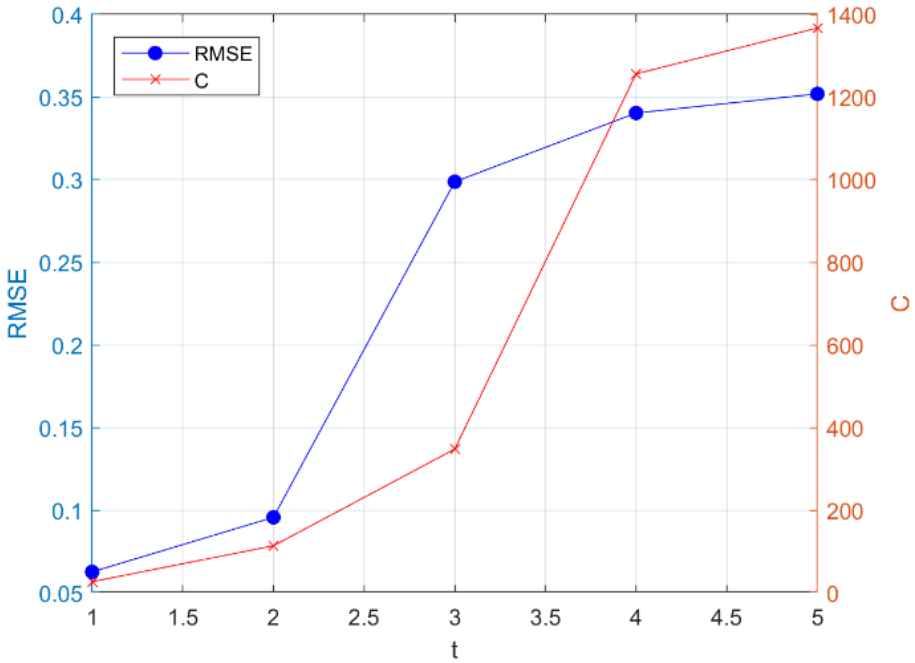

Figure 1. Visualization of Table 1 Data

The observed trends in the curves indicate that as t increases, both RMSE error and computational complexity C also increase, but eventually tend to stabilize. This indicates that the matrix decomposition approach, incorporating the improved evolutionary algorithm under sparsity constraints, is feasible for matrices of relatively smaller dimensions.

The algorithm also yields optimized solutions for the DFT matrix F_N at different t values (i.e., t=1,2,3,4,5). Taking t=2 (i.e., N=4) as an example, the optimized values β=0.0032, the minimum RMSE = 0.0957, and hardware complexity C=114. The decomposition matrix $\mathcal{A} = \{A_1, A_2, \ldots, A_K\}$, where K=4, is as follows:

$$A_1=\begin{pmatrix} -4+4i & 0 & 0 & -1-4i \\ 0 & -1+i & 0 & -2+i \\ 2-2i & 0 & -1+2i & 0 \\ 2-4i & 0 & -1+2i & 0 \end{pmatrix}, A_2=\begin{pmatrix} 1+i & 0 & -1+2i & 0 \\ 4+2i & 0 & 1-2i & 0 \\ 1+4i & 0 & -2-i & 0 \\ -2-4i & 0 & 0 & -1-i \end{pmatrix},$$
$$A_3=\begin{pmatrix} 0 & 0 & 0 & -4 \\ 0 & -2-i & 0 & 2+2i \\ 0 & 0 & 0 & -1-2i \\ 1-2i & 0 & 0 & -2+i \end{pmatrix}, \quad A_4=\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{19}$$

Additionally, for t=5, i.e., N=32, the RMSE error variation curve as the number of iterations increases is shown below:

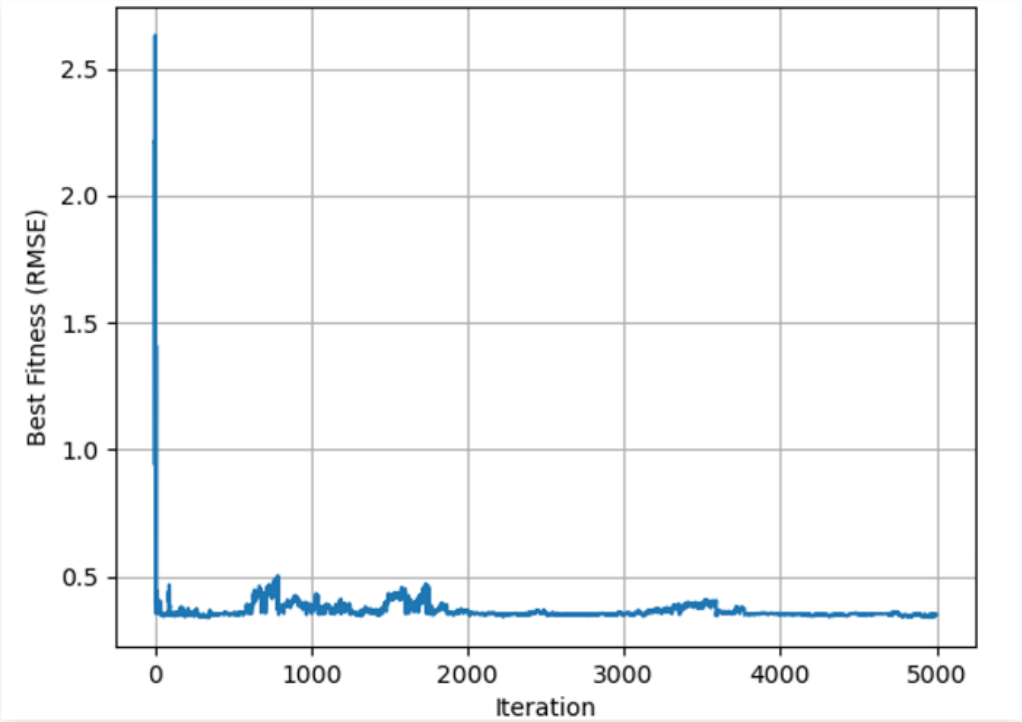

Figure 2. Best fitness over iterations

The figure shows that as the number of iterations increases to around 2000, the RMSE error shows minimal change, with the final error stabilizing around 0.35. This indicates that the improved evolutionary algorithm has converged, thereby proving that the approach of using integer matrix decomposition with sparsity constraints is feasible. This result was obtained under certain precision conditions.

To further compare the differences between integer matrix decomposition approximation algorithm with sparsity constraints and traditional FFT algorithms, this paper conducts a comparative analysis of the computational complexity of the three FFT algorithms mentioned earlier: (i) Cooley-Tukey FFT [5]; (ii) Split-Radix FFT [8]; and (iii) Winograd FFT [9], under the condition t=5, i.e., N=32 (for a DFT matrix of size 32×32). The experimental results are as follows:

Table 2. Comparison and Analysis of Algorithm Complexity

Algorithm	RMSE	C
Cooley-Tukey FFT	0	2302
Split-Radix FFT	0	2797
Winograd FFT	0	3313
Algorithm 1	0.3519	1367

Since traditional FFT algorithms are based on the exact computation of the DFT matrix, their RMSE error is always 0. However, from the perspective of computational complexity, although our proposed Algorithm 1 sacrifices some degree of accuracy, it minimizes the complexity. In practical engineering applications, where the hardware requirements are not stringent, Algorithm 1 offers broader engineering applicability. Additionally, based on the previously proposed **Algorithm 1** and **Algorithm 2**, the numerical experiment results for the DFT matrix F_N with N=1024, obtained using Python programming, are as follows:

Table 3. Error and Complexity of Matrix Decomposition under Kronecker Product

RMSE	*C*	$RMSE_1$	$RMSE_2$	β_1	β_1
0.17678	501038	0.018242	0.19291	0.036483	0.545632

Among them, $RMSE_1, RMSE_2, \beta_1, \beta_2$ represent the error values and scaling factor parameters obtained after the decomposition of the two DFT matrices F_{N1} and F_{N2} for N=32. To further compare and analyze the differences between our proposed matrix decomposition algorithm based on the divide-and-conquer strategy using the Kronecker product and the traditional FFT algorithm, this study conducts a computational complexity comparison under the condition t=10, i.e., $N = 2^t = 1024$ (at which point the DFT matrix is of size 1024×1024). The experimental results are as follows:

Table 4. Comparison and Analysis of Algorithm Complexity

Algorithm	**RMSE**	**C**
Cooley-Tukey FFT	0	1051289
Split-Radix FFT	0	2009784
Winograd FFT	0	2301357
Algorithm 2	0.17678	501038

The results in **Table 4** indicates that as the DFT matrix dimension N increases, although the error for the traditional FFT algorithm remains 0, its computational complexity C grows quadratically with the DFT matrix dimension N. In contrast, the computational complexity of the method proposed in this paper is C=501038. This reduction in complexity is due to the decomposition of the original F_N into two DFT matrices F_{N1} and F_{N2} using the Kronecker product. As a result, the computational complexity is several times lower than that of the traditional FFT algorithm. Furthermore, the root mean square error (RMSE) is 0.17678, which is relatively small. By sacrificing a certain level of precision, the proposed method demonstrates good optimization under the constraint conditions. Additionally, in the actual decomposition process, the execution speed is significantly faster than directly decomposing the matrix F_N, clearly demonstrating the superiority of the method proposed in this paper.

4. Conclusions

Currently, using the FFT algorithm for DFT calculations results in relatively high hardware complexity. Therefore, based on prior work, this paper proposes an alternative approach to reduce the hardware complexity of DFT calculations. The core idea of this approach is to approximate the DFT matrix as a product of sparse matrices, with elements constrained to a limited range. This is a trade-off between precision and hardware complexity, in which some calculation accuracy is sacrificed to significantly reduce hardware complexity. Based on this idea, this paper first investigates the integer matrix decomposition approximation algorithm under sparsity constraints. Using an improved evolutionary algorithm, the DFT matrices F_N for $N = 2^t, t = 1,2,3,4,5$, are optimized for A and β, satisfying constraint conditions 1 and 2. The experimental results demonstrate that the method performs well compared to the traditional FFT algorithm.

In addition, this paper focuses on the problem of large-scale DFT matrix decomposition. To address this, the paper proposes a matrix decomposition approximation based on the Kronecker product. Using the divide-and-conquer strategy, the large-scale DFT matrix $\mathbf{F}_N = \mathbf{F}_{N1} \otimes \mathbf{F}_{N2}$ are DFT matrices of dimensions N_1 and N_2, respectively, and $\otimes$ represents the Kronecker product, was decomposed into smaller matrices for processing. The numerical experimental results show that this method reduces the computational complexity by approximately 5 times compared to the traditional FFT algorithm, highlighting its significant advantages and practical engineering value.

References

[1] Gao, Z., Wang, Q., Chen, A., Liu, Z., Wu, B., Chen, L., et al. Parameter-efficient fine-tuning with discrete Fourier transform [J]. Proceedings of the International Conference on Machine Learning, 2024.

[2] Xu, W., Zhang, Z. An accurate discrete Fourier transform to analyse the absolute phase of THz pulses [J]. Journal of Optics, 2024, 26(5): 055202.

[3] Madanayake, A., Cintra, R. J., Akram, N., Ariyarathna, V., Mandal, S., Coutinho, V. A., Bayer, F. M., & Coelho, D. Fast radix-32 approximate DFTs for 1024-beam digital RF beamforming [J]. IEEE Access, 2020, 96613-96627.

[4] Chen, H., Liu, Y., & Gao, F. (2022). Approximate Matrix Decomposition for Low-Complexity Beamforming in Massive MIMO Systems. IEEE Transactions on Wireless Communications, 21(5), 3456-3468.

[5] Chen, Y., & Patel, R. Advancements in Fast Fourier Transform Techniques for Real-Time Applications [J]. 2022.

[6] Wang, M., Fu, X., Yan, X., Teng, L. A new chaos-based image encryption algorithm based on discrete Fourier transform and improved Joseph traversal [J]. Mathematics, 2024, 12(5): 638.

[7] Mavridis, G. A., Kolios, A. J. Comprehensive performance evaluation of computationally efficient discrete Fourier transform structures [J]. IEEE Transactions on Signal Processing, 2023, 71: 1001-1015.

[8] Zhang, Y., Chen, L. Real time method and algorithms for fast discrete Fourier transform of discrete finite signals [C]. IEEE Conference Publication, 2024.

[9] Zhang, Z., Zhang, P., Xu, Z., Yan, B., & Wang, Q. Im2col-Winograd: An Efficient and Flexible Fused-Winograd Convolution for NHWC Format on GPUs [J]. Proceedings of the 53rd International Conference on Parallel Processing (ICPP'24), 2024.

[10] Nanthakumar, Wijenayake, Edussooriya, C. U. S., & Madanayake, A. (2023). Combined approximate transforms and approximate computing for low-complexity multibeam arrays [C]. In 2023 22nd International Symposium on Communications and Information Technologies (ISCIT), Sydney, Australia, 2023, pp. 145-150.

[11] Kumar, A., Singh, P., & Gupta, R. (2024). Hybrid analog-digital beamforming with low-complexity algorithms for 5G systems. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 8(3), 466-479.

[12] Keles, D. A., & Sahin, O. (2023). Massive MIMO technology: A key advancement in shaping communication networks for the 5G/5G+ era [C]. In 2023 IEEE International Conference on Communications (ICC), pp. 1-6.

[13] Zhou, Y., Cao, W., Liu, L., Agaian, S., & Chen, C. L. P. (2022). Fast Fourier transform using matrix decomposition [J]. Information Sciences, 291, 172-183.

[14] Mohan, P. Genetic algorithms: Theory, genetic operators, solutions, and applications [J]. Applied Intelligence, 2023, 53(3): 1456-1474.

[15] Cohen, I., Huang, G., & Benesty, J. (2023). Kronecker-product beamforming with sparse concentric circular arrays [J]. IEEE Transactions on Signal Processing, 71, 123-134.

[16] Stansbury, C., Pickard, J., & Rajapakse, I. (2024). Hypergraphs, tensors, and the Kronecker product [J]. Applied Mathematics and Computation, 500, 125340.