*electronics*

MDPI

*Article*

# Improved Multi-Strategy Matrix Particle Swarm Optimization for DNA Sequence Design

**Wenyu Zhang [1], Donglin Zhu [2], Zuwei Huang [3] and Changjun Zhou [2,*]**

[1] Faculty of Information Science and Engineering, Ocean University of China, Qingdao 266100, China
[2] College of Mathematics and Computer Science, Zhejiang Normal University, Jinhua 321004, China
[3] School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China
* Correspondence: zhouchangjun@zjnu.edu.cn

**Abstract:** The efficiency of DNA computation is closely related to the design of DNA coding sequences. For the purpose of obtaining superior DNA coding sequences, it is necessary to choose suitable DNA constraints to prevent potential conflicting interactions in different DNA sequences and to ensure the reliability of DNA sequences. An improved matrix particle swarm optimization algorithm, referred to as IMPSO, is proposed in this paper to optimize DNA sequence design. In addition, this paper incorporates centroid opposition-based learning to fully preserve population diversity and develops and adapts a dynamic update on the basis of signal-to-noise ratio distance to search for high-quality solutions in a sufficiently intelligent manner. The results show that the proposal of this paper achieves satisfactory results and can obtain higher computational efficiency.

**Keywords:** DNA computing; DNA sequences design; improved matrix particle swarm optimization algorithm (IMPSO); opposition-based learning; signal-to-noise ratio distance

check for updates

**Citation:** Zhang, W.; Zhu, D.; Huang, Z.; Zhou, C. Improved Multi-Strategy Matrix Particle Swarm Optimization for DNA Sequence Design. *Electronics* **2023**, *12*, 547. https://doi.org/10.3390/electronics12030547

Academic Editor: Janos Botzheim

Received: 24 December 2022
Revised: 16 January 2023
Accepted: 18 January 2023
Published: 20 January 2023



## 1. Introduction

DNA is a macromolecular polymer composed of deoxyribonucleotides, which are composed of deoxyribose, phosphate and bases including adenine (A), guanine (G), thymine (T) and cytosine (C). In 1953, after experimentational analysis, Watson and Crick proposed a molecular model of the double-helix structure of DNA [1] and first proposed the principle of base complementary pairing, in which the bases of the nucleotide residues in a nucleic acid molecule are linked to each other by hydrogen bonds in the correspondence between A and T and G and C. That is to say four possible base pairs for the $A = T$, $T = A$, $G \equiv C$ and $C \equiv G$. A and T form two hydrogen bonds between; G and C constitute the three hydrogen bonds between. In 1994, Turing Award-winner Adleman [2] proposed a simple problem computation using the principle of the base complementary pairing of DNA, thus inaugurating DNA computing. DNA computing then continued to evolve toward generalization. In 2006, Winfree [3] proposed the DNA strand replacement reaction, which was a new way to construct logic circuits. In addition to circuit computing, DNA computing can be combined with a variety of intelligent computing methods, such as neural network chaotic systems, and used in different fields.

According to the biological composition of DNA, DNA can be considered a long string of four symbols, they are A, G, C and T. Through the alphabet of $\sum = \{A, G, C, T\}$, two binary numbers or one quadratic number can be used to encode DNA to store information. In 2012, Church [4] led the first team to store a book of 659 kb in DNA, demonstrating the storage capacity of DNA. In 2016, Extance [5] showed that 1 g of DNA can hold the contents of 100 billion DVDs and that 1 kg of DNA can even hold all the information data in the world. In the same year, Zhirnov et al. [6] found that DNA information storage density is 10 million terabytes per cubic centimeter and that even simple E. coli have a storage density of about 1019 bits per cubic centimeter, further validating the powerful storage capacity of DNA. In addition, due to the inherent parallel mechanism of DNA, i.e., the

phenomenon that the leading strand and the trailing strand are replicated simultaneously, DNA computation can be performed simultaneously on many DNA strands, which greatly enhances the speed of DNA computation.

DNA coding sequence design is a key step in DNA computation, which realizes the computation and transformation of data stored in it through specific reactions between DNA molecules. The rationality of DNA coding is directly related to whether the model can be successfully validated by biochemical experimentations and the accuracy of DNA computation. However, DNA encoding needs to satisfy molecular biology constraints, including physical constraints such as GC content constraints and thermodynamic constraints such as melting temperature (Tm).

Efficient DNA computation cannot be carried out without excellent DNA coding. Optimal DNA coding can be obtained by optimal coding algorithms, but the cost required for optimal coding may not be satisfied in a large problem space. Therefore, in order to provide efficient and suitable DNA coding in acceptable computational time and space, heuristic algorithms are widely applied to the design of DNA sequences in recent years as a shortcut algorithm. Zhu et al. [7] proposed an IBPSO algorithm to solve the DNA sequence design problem, as well as further improving the quality of DNA sequences. Chaves-González et al. [8] fused artificial bee colony algorithms to propose a new evolutionary approach to create a DNA sequence on the strength of multi-objective swarm intelligence to automatically generate reliable DNA strands that can be applied to molecular computing. Yang et al. [9] improved the spatial dispersion in the traditional IWO algorithm and used the IWO algorithm and the niche crowding in the algorithm to solve the DNA sequence design problem. Zhang et al. [10] used an improved taboo search algorithm for improving the means for the systematic design of equal-length DNA strands, which conduces the discovery of a range of good DNA sequences that satisfy the required certain combinatorial and thermodynamic constraints. Cervantes-Salido et al. [11] proposed a multi-objective evolutionary algorithm for designing a DNA sequence, taking advantage of a matrix-based GA along with specific genetic operators to improve the performance for DNA sequence optimization compared to previous methods. Chaves-González et al. [12] proposed an adapted multi-objective version of the differential evolution (DE) metaheuristics approach incorporating a multi-objective standard fast non-dominated sorting genetic algorithm to produce high-quality DNA sequences. Vega-Rodríguez et al. [13] made several rectifications in the noted fast non-dominated sorting genetic algorithm in conjunction with a novel multi-objective algorithm in accordance with the behavior of fireflies and proposed a new DNA sequence design method based on multi-objective firefly algorithm for generating reliable DNA sequences for molecular computing. The metaheuristic algorithm as a general heuristic algorithm can greatly reduce the number of attempts in a limited searching space, can achieve the problem solution rapidly and is heavily applied to generate reliable DNA coding sequences by virtue of its high efficiency. However, metaheuristic algorithms, as a product of combining random algorithms with local search algorithms, are susceptible to randomness or fall into a local optimum due to premature search and do not necessarily guarantee the feasibility and reliability of the resulting DNA sequences. In recent years, in order to improve the metaheuristic algorithm, which is prone to being caught in a local optimality, many scholars have done a lot of corresponding research and proposed various improved metaheuristic algorithms, among which the particle swarm algorithm is a theoretically mature and widely used emerging metaheuristic algorithm to find the optimal solution through collaboration and information-sharing among individuals in the population.

Particle swarm optimization [14] (PSO) is a method to seek out the global optimum by following the current searched optimum based on the observation of the regular behavior of the flock activity. This algorithm has appealed to the academics with the strong points of easy implementation, high-accuracy and fast convergence and has shown advantages in solving practical problems. However, if the parameters are not chosen reasonably, the particles may miss the optimal solution and subsequently appear to be non-converging.

Even if all particles move in the direction of convergence, homogenization can occur. Due to the loss of the diversity of the population in the search space, premature convergence, poor local search ability, etc., can occur, leading to a lack of further improvement in accuracy as well as falling into a local optimum. In specific problems, the PSO needs to be analyzed and improved in order to achieve better results. Houssein et al. [15] experimentally demonstrated that the PSO algorithm suffers from premature convergence, being trapped in a local optimum and poor performance in multi-objective optimization. Ghatasheh et al. [16] used innovative optimization paradigms to improve the prediction power of bankruptcy modeling to generate prediction models. Zhang et al. [17] proposed a new vector co-evolutionary particle swarm optimization algorithm (VCPSO) to enhance population diversity and avoid premature convergence, but it suffers from falling into local optima or inefficient execution. The multi-objective particle swarm optimization algorithm (MOPSO) proposed by Coello et al. [18] has good search performance but only focuses on the generation of non-dominated vectors and maintaining population diversity, without considering the constraint functions. The region-based selection algorithm (PESA-II) in evolutionary multi-objective optimization proposed by Corne et al. [19] shows outstanding performance in region-based selection multi-objective algorithms but does not deal with runtime complexity. Eberhart et al. [20] used a dynamic neighborhood particle swarm optimization approach to solve multi-objective optimization problems, which is easy to implement and requires few parameters to be tuned but only deals with unconstrained multi-objective optimization problems. Deb et al. [21] developed a fast and elitist multi-objective genetic algorithm (NSGA-II) based on multi-objective evolutionary algorithm (MOEA), which is able to find better solution diffusion and better convergence for most of the problems but NSGA-II algorithm uses the no-penalty parameter constraint processing method, which has some limitations.

In this study, an improved multi-strategy matrix particle swarm-based optimization algorithm, referred to as IMPSO, is proposed. Compared with the previous matrix particle swarm algorithm, the running time under the same conditions is significantly reduced and the values of the constraints on the DNA sequences are well maintained. In addition, centroid opposition-based learning strategy is incorporated to preserve population diversity and to obtain global and sufficient results; at the same time, this strategy is used to reinitialize the population when the iteration numbers is a multiple of 100 to prevent the algorithm falling into the local optimal solution, while a dynamic update in accordance with signal-to-noise ratio distance is developed and adapted to search for high-quality solutions in a sufficiently intelligent manner and enable every individual to search for the best position within its own near neighborhood. The application of these two strategies puts the global optimal solution into effect. What is more, suitable DNA constraints are chosen to avoid potential conflicting interactions between DNA molecules to prevent the generation of secondary structures, to control non-specific hybridization and to ensure the reliability of DNA sequences. To verify the feasibility of the IMPSO algorithm, the DNA sequences, the values of each constraint and their running times obtained from the optimization of IMPSO with MPSO [22], IWO [23], PSO [24] and HS [25] were compared. MPSO continues the search processes by introducing the speed and position update mechanism of the global best particle, effectively ensuring the convergence. IWO is a simple but effective algorithm employed for finding a solution for an engineering problem. PSO is a typical SI that reproduces the new population by learning from personal and global guidance information. HS is a optimization algorithm to solve TSP and a specific academic optimization problem, etc., by mimicking the improvisation of music players. To show the competitiveness of the IMPSO algorithm in solving the DNA sequence design problem, this paper compares the experimentational DNA sequence design results of IMPSO with those of NCIWO, HSWOA [26], MO-ABC, CPSO [27] and DMEA [28]. NCIWO and MO-ABC are mentioned above when introducing particle swarm optimization. HSWOA [26] is used to design DNA sequences that meet the new combination constraint. CPSO [27] is used to solve precocious phenomena and the local optimum of PSO by chaotic mapping.

DMEA [28] is proposed to solve the DNA sequences design and to mitigate an NP-hard problem. With the same number of iterations, the experimental results show that the scheme is more competitive and has higher computational efficiency in solving the DNA sequence design problem. The main contributions of this study are as follows:

(1) The matrix particle swarm optimization is introduced to improve the efficiency of the traditional PSO.
(2) On the basis of the centroid opposition-based learning strategy, the influence of the optimal and worst position is considered to make the position update more reasonable.
(3) The concept of signal-to-noise ratio distance is led into, and a formula conforming to the internal state of the population is designed.
(4) During DNA sequence optimization design experimentation, the rationality and effectiveness of IMPSO are verified by comparing with the variations of various algorithms.

The rest of the paper is arranged in the following way. Section 2 presents the constraints associated with designing DNA coding sequences. Section 3 describes the strategy along with the algorithm flow of the IMPSO. Section 4 introduces the comparison and analysis of the IMPSO algorithm with other optimization algorithms for DNA sequence design. Section 5 outlines the conclusions of this paper and indicates the next steps.

## 2. Constraints Formulation for DNA Sequence Design

Reliable DNA sequence design is a two-dimensional discrete optimization problem, and the relevant constraints can be partitioned into two categories, one is the combination of constraints including continuity, hairpin, H-measure and similarity, aiming to improve the specificity of DNA molecule recognition, and the other is thermodynamic constraints, mainly including melting temperature (Tm) and free energy, aiming to ensure the consistency of the physicochemical properties of DNA molecules.

This section describes in detail the constraints associated with designing DNA sequences. In the following constraint equation, $S$ stands for the DNA sequence set; $u$ and $v$, respectively, represent two certain DNA sequences selected from the DNA sequence set $S$; $\alpha$ is the DNA sequences number contained in DNA sequence set $S$, and $\beta$ is the number of bases contained in a given DNA sequence in $S$. $T(a,\ T_{value})$ is a threshold function that returns a when the value is $a > T_{value}$, and 0 otherwise. If $u$ and $v$ are complementary, the function $cd(u,\ v)$ returns 1; otherwise, the result of the equation is 0.

### 2.1. Continuity

Continuity is the amount of contiguous identical bases (A,C,G,T) in a given single strand of DNA. Too large a continuity value in the DNA sequence makes the DNA sequence easily twisted and folded in the hybridization process, thus creating a secondary structure that is not conducive to DNA computation. Assuming the continuity threshold is 3, for the DNA sequence CAATGCGTTAGCCCCGATCTTAC, it reaches the continuity threshold, after which the sequence will use the continuity function to calculate its continuity value, and other sequences that do not trigger the threshold will be considered discontinuous. The formula to calculate the continuity of a certain DNA strand is as shown below [12].

$$f_{continuity}(S) = \sum_{\rho=1}^{\alpha} Continuity(S_\rho) \tag{1}$$

$$Continuity(u) = \sum_{i=1}^{\beta-CT} T(cont_\sigma(u,i), CT) \tag{2}$$

$$cont_\sigma(u,i) = \begin{cases} \theta, & if\ \exists\theta\ s.t. u_i \neq \sigma, u_{i+\theta+1} \neq \sigma, u_{i+j} = \sigma\ for\ 1 < j \leq \theta \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$\sigma \in \{A, G, C, T\}$; $CT$ is the threshold value; $T(A, CY)$ is a count of the number of contiguous bases in DNA above a threshold; if $A > CY$, then return $A$; otherwise, return 0. $cont_\sigma(u, i)$ returns the number of consecutive bases of sequence $u$.

### 2.2. Hairpin

During the process of DNA sequence self-hybridization, the overlapping part of the sequence will fold and the corresponding bases will complementarily pair, and the pairing forms a secondary structure called a hairpin structure. The hairpin structure consists of a hair stem and a hair loop. If the hairpin structure is present in the DNA sequence, it will undergo self-folding in the biochemical reaction. For avoiding self-hybridization in DNA sequences, making the hairpin structure in DNA sequences as small as possible is of great importance. There are two types of hairpin structures, hair stem and hair loop. $L_{min}$ is the minimum hair loop length required for the hairpin structure; $T_{min}$ is the minimum hairpin stem length required; $l$ is the length of the hair loop; $t$ is the length of the hair stem, and the formula to calculate a DNA hairpin is as shown below [12].

$$f_{hairpin}(S) = \sum_{\rho=1}^{\alpha} Hairpin(S_\rho) \tag{4}$$

$$Hairpin(u) = \sum_{t=T_{min}}^{(\beta - L_{min})} \sum_{l=L_{min}}^{\beta-2t} \sum_{i=1}^{\beta-2t-l} T(\sum_{j=1}^{PL_{til}} cb(u_{i+j}, u_{\beta-j}), \frac{PL_{til}}{2}) \tag{5}$$

where $PL_{til} = \min(t + i, \beta - l - i - t)$ represents the maximum number of base pairs possible when $t + i + \frac{l}{2}$ is the center of the hairpin structure. $cb(u, v)$ determines whether $u$ and $v$ are complementary; if $u$ and $v$ are complementary, the result is 1; otherwise, the result is 0.

### 2.3. H-Measure

In DNA sequences, *H-Measure* is adapted to count the Hamming distance, which indicates the number of different bases at the same position of two complementary DNA sequences. The likelihood of hybridization between complementary strands of the same DNA molecule is closely linked to the *H-Measure*, showing a positive correlation. With this constraint, non-specific hybridization between a DNA sequence and its complementary sequences can be controlled. *H-Measure* is calculated by the following formula [12].

$$f_{H-measure}(S) = \sum_{\rho=1}^{\alpha} \sum_{\theta=1, \rho \neq \theta}^{\alpha} H - measure(S_\rho, S_\theta) \tag{6}$$

where $S_\rho, S_\theta$ respectively represent two reverse parallel DNA sequences. *H-Measure* calculation consists of two parts: continuous and discontinuous calculations.

$$\begin{aligned} H - measure(u, v) = Max_{g,t}(h_{dis}(u, FShift(v(-)^g v, t)) \\ + h_{cont}(u, FShift(v(-)^g v, t)) \end{aligned} \tag{7}$$

$$h_{dis}(u, v) = T(\sum_{i=1}^{\beta} cb(u_i, v_i), DH \times \beta) \tag{8}$$

$$h_{cont}(u, v) = \sum_{i=1}^{\beta} T(subcb(u, v, i), CH) \tag{9}$$

$h_{dis}(u, v)$ calculates the number of complementary bases in the DNA sequence $u, v$. $h_{cont}(u, v)$ figures the penalty value of the consecutive base pairing of DNA sequences $u$ and $v$. $v(-)^g v$ is a sequence formed by splicing two fragments of sequence $v$ with a splice gap of $g$. *H-Measure* is the maximum value after the summation of the above two functions. $subcb(u, v, i)$ defines the number of consecutive complementary paired bases of the $u, v$

sequence to begin with position $i$. $DH$ is a real number in [0, 1], and $CH$ is a positive integer in [1, $N$].

### 2.4. Similarity

In DNA calculations, similarity indicates how close two DNA sequences are to each other in terms of bases at the same position. Similarity takes into account the complementary Hamming distance after shifting in addition to the Hamming distance. The similarity value is the maximum value of the totality of the amount of bases with the same displacement and the amount of consecutive identical bases between sequences $u$ and splicing sequence $v(-)_g v$. The similarity is calculated as follows [12].

$$f_{similarity}(S) = \sum_{\varepsilon=1}^{\alpha} \sum_{\delta=1, \varepsilon \neq \delta}^{\alpha} Similarity(S_\varepsilon, S_\delta) \tag{10}$$

where $S\varepsilon$, $S\delta$ denotes two sequences in the DNA sequence set $S$. The similarity is calculated in two parts: the similarity of discontinuous sequences and the similarity of the largest continuous common subset.

$$\begin{aligned} Similarity(u, v) = Max_{g,t}(s_{dis}(u, FShift(v(-)^g v, t)) \\ + s_{cont}(u, FShift(v(-)^g v, t))) \end{aligned} \tag{11}$$

$$s_{dis}(u, v) = T(\sum_{i=1}^{\beta} eq(u_i, v_i), DS \times \beta) \tag{12}$$

$$s_{cont}(u, v) = \sum_{i=1}^{\beta} T(subeb(u, v, i), CS) \tag{13}$$

$FShift(v(-)^g v, t)$ denotes the shift of $v(-)^g v$ by $t$ positions, $eq(u,\ v)$ is used to determine whether $u$ and $v$ are equal; equal returns 1; otherwise, the result is 0; $DS$ is a real number in [0, 1], and $CS$ is a positive integer in [1, $N$]. $subeb(u,\ v,\ i)$ shows the amount of consecutive equal bases from DNA sequence $u$ and $v$ starting from position $i$. $S_{dis}(u, v)$ calculates the Hamming distance of two DNA strands; $S_{cont}(u, v)$ calculates the sum of the consecutive equal numbers of bases starting from positions 1 to $\beta$.

### 2.5. GC Content [29]

GC content stands for the amount of guanines as well as cytosines in the DNA sequence as a percentage of the overall number of bases. GC content is directly related to the biochemical stability of DNA sequences because $G \equiv C$ base pairs contain three hydrogen bonds and release more heat energy when broken than $A = T$ base pairs containing two hydrogen bonds, so GC content also influences the melting temperature of DNA sequences. For the DNA sequence ACGTCGTTCGTACGC, the GC content is 60% (9/15). The GC content (in percentage form) is calculated by the following formula.

$$GC(u) = 100 \sum_{i=1}^{\beta} \frac{GC(u_i)}{\beta} \tag{14}$$

$$GC(\tau) = \begin{cases} 1, & \tau = G \ or \ \tau = C \\ 0, & \tau = A \ or \ \tau = T \end{cases} \tag{15}$$

### 2.6. Melting Temperature (Tm)

Melting temperature is the temperature required for half of the base pairs of a DNA double-stranded structure to be disrupted into a single-stranded structure. Melting temperature is an important thermodynamic constraint of DNA molecules that influences the reaction efficiency of DNA sequences, and a steady Tm allows for the better control of hybridization reactions between DNA molecules. The $G \equiv C$ base pair contains three

hydrogen bonds and releases more thermal energy upon breaking than the $A = T$ base pair containing two hydrogen bonds. Tm is usually calculated in accordance with the nearest-neighbor thermodynamic model [30], with the following relevant equation.

$$f_{T_m}(S) = \frac{\Delta H^{\circ}}{\Delta S^{\circ} + R \ln\left(\frac{[C_T]}{4}\right)} - 273.15 \tag{16}$$

where $\Delta H^{\circ}$ represents the enthalpy change from reactants to products, which is the total enthalpy of adjacent bases; $\Delta S^{\circ}$ represents the entropy change from reactants to products, which is the total entropy of adjacent bases. $R$ represents the gas constant (1.987 cal/kmol), and $C_T$ is the concentration of DNA molecules.

### *2.7. Fitness Function*

The optimization problem of this paper belongs to the minimum optimization problem. The fitness function of the DNA sequence is determined by the constraint function described above and is the minimum of the above constraint functions, expressed by the following formula.

$$\begin{gathered} Minimize\ f_i(x), i \in \{Continuity, Hairpin, H-measure, \\ Similarity\}\ subject\ to\ GC = 50\%,\ Tm \end{gathered} \tag{17}$$

## 3. Improved Multi-Strategy Matrix Particle Swarm Optimization

### *3.1. Basic Information of Matrix Particle Swarm*

In order to describe the IMPSO algorithm more clearly, this section first introduces information about matrix particle swarm, some important formulas used by the algorithm and the operations between matrices.

#### 3.1.1. Representation Information

Assume there exists a $N$ individuals population to solve the $D$-dimensional problem. This population is represented by a matrix $X$ of size $N \times D$, defined as follows.

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{pmatrix} \tag{18}$$

where $x_{ij}$ represents the individual $i$ and dimension $j$.

To accommodate the matrix-based representation, the upper bound of the variables is represented by a matrix $XB$ of size $1 \times D$, the lower bound of the variables is represented by a matrix $XM$ of size $1 \times D$, and the fitness values of every individual are represented by a matrix $Fit$ of size $N \times 1$. The matrix $Ones$ is an all-1 matrix, and the matrix $R$ is a matrix consisting of random numbers of [0, 1].

#### 3.1.2. Common Matrix Operations

Table 1 lists the relevant matrix operations used in this paper and shows their corresponding descriptions. For convenience of description, the size of matrices $A$ and $B$ defaults to $N \times D$ if not specifically mentioned.

#### 3.1.3. Initialization of Particle Swarm Related Variables

Matrix $X$, also called the population matrix, represents the position of individuals. Matrix $V$ represents the velocity, and $pBest$ represents the personal best positions of all the individuals in the population, respectively. Where $X$ is initialized as follows.

$$X_{N\times D} = Ones_{N\times 1} \times (XB - XM) \circ R_{N\times D} + Ones_{N\times 1} \times XM \tag{19}$$

**Table 1.** Typical operations in matrix and their notations [22].

| Name | Description |
| --- | --- |
| Addition operation $(+)$ | $A+B=\begin{pmatrix} a_{11}+b_{11} & \cdots & a_{1D}+b_{1D} \\ \vdots & \ddots & \vdots \\ a_{N1}+b_{N1} & \cdots & a_{ND}+b_{ND} \end{pmatrix}$ |
| Subtraction operation $(-)$ | $A-B=\begin{pmatrix} a_{11}-b_{11} & \cdots & a_{1D}-b_{1D} \\ \vdots & \ddots & \vdots \\ a_{N1}-b_{N1} & \cdots & a_{ND}-b_{ND} \end{pmatrix}$ |
| Multiplication operation $(\times)$ | $A_{N\times D}\times B_{D\times N}=\begin{pmatrix} \sum_{i=1}^{D} a_{1i}\times b_{i1} & \cdots & \sum_{i=1}^{D} a_{1i}\times b_{iN} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{D} a_{Ni}\times b_{i1} & \cdots & \sum_{i=1}^{D} a_{Ni}\times b_{iN} \end{pmatrix}$ |
| Scalar multiplication $(\cdot)$ | $c\cdot A=\begin{pmatrix} c\times a_{11} & \cdots & c\times a_{1D} \\ \vdots & \ddots & \vdots \\ c\times a_{N1} & \cdots & c\times a_{ND} \end{pmatrix}$ |
| Hadamard product $(\circ)$ | $A\circ B=\begin{pmatrix} a_{11}\times b_{11} & \cdots & a_{1D}\times b_{1D} \\ \vdots & \ddots & \vdots \\ a_{N1}\times b_{N1} & \cdots & a_{ND}\times b_{ND} \end{pmatrix}$ |
| Transposition operation $\left(X^T\right)$ | $A^T=\begin{pmatrix} a_{11} & \cdots & a_{D1} \\ \vdots & \ddots & \vdots \\ a_{1N} & \cdots & a_{DN} \end{pmatrix}$ |
| Logical operation $(\leq)$ | $A\leq B=C,\ c_{i,j}=\begin{cases} 1, & if\ a_{i,j}\leq b_{i,j} \\ 0, & otherwise \end{cases}$ |
| Maximum operation $(max)$ | $a=max(A)$, where a is the maximum element in A |
| Minimum operation $(min)$ | $a=min(A)$, where a is the minimum element in A |
| Maximum indexing $(maxind)$ | $k=maxind(A_{N\times 1})$, where k is the row index of the maximum element in $A_{N\times 1}$ |
| Minimum indexing $(minind)$ | $k=minind(A_{N\times 1})$, where k is the row index of the minimum element in $A_{N\times 1}$ |
| Index operation $(X[I\mid J])$ | $X[I\mid J]=\begin{pmatrix} X_{I_1J_1} & \cdots & X_{I_1J_j} \\ \vdots & \ddots & \vdots \\ X_{I_iJ_1} & \cdots & X_{I_iJ_j} \end{pmatrix}$ |

The initialization process of $V$ is as follows.

$$V_{N\times D}=Ones_{N\times 1}\times(VB-VM)\circ R_{N\times D}+Ones_{N\times 1}\times VM \tag{20}$$

After the initialization of matrices $X$ and $V$ is completed, IMPSO obtains the fitness values of all individuals, represented by a matrix $Fit$ of size $N\times 1$, according to the following equation.

$$Fit_{N\times 1}=f(X) \tag{21}$$

The initialization process of $pBest$ is as follows.

$$pBest_{N\times D}=Ones_{N\times 1}\times(XB-XM)\circ R_{N\times D}+Ones_{N\times 1}\times XM \tag{22}$$

The initialization process of *pBest_Fit* is as follows.

$$pBest\_Fit_{N\times D} = Ones_{N\times 1} \times (VB - VM) \circ R_{N\times D} + Ones_{N\times 1} \times VM \tag{23}$$

After completing the above variable initialization process, the globally best fitness value can be obtained by the following formula, represented by *gBest_Fit*.

$$gBest\_Fit = \begin{cases} \min(Fit), & if\ it\ is\ a\ minimum\ problem \\ \max(Fit), & if\ it\ is\ a\ maximum\ problem \end{cases} \tag{24}$$

Furthermore, the optimization problem considered in this experimentation is the minimum value problem; IMPSO can use *minind()* formula in Table 1 to obtain the corresponding number of rows for individuals with the best *pBest* fitness value, as follows.

$$Index = minind\ (pBest\_Fit) \tag{25}$$

3.1.4. Velocity and Position Update

In the process of IMPSO iterations, the population continuously performs velocity update as well as position updates from generation to generation in order to get as close as possible to the global optimum, and the equations for velocity and position updates are shown below.

$$V = \omega \times V + c_1 \times R_1 \circ (pBest - X) + c_2 \times R_2 \circ (Ones \times gBest - X) \tag{26}$$

$$X = X + V \tag{27}$$

It is worth noting that the matrix *gBest* of size $1 \times D$ is actually the individual with the best fitness value in the matrix *pBest* of $N \times D$, which is the index row corresponding to *pBest*. The $N \times D$ matrix $X$ extended from the $1 \times D$ matrix *gBest* can be obtained by the following matrix multiplication formula, which shows that the value of each row of the matrix $X$ is equal to the value of *gBest*.

$$X_{N\times D} = Ones_{N\times 1} \times gBest_{1\times D} \tag{28}$$

In order to avoid the elements of matrices $V$ and $X$ to exceed the space boundary, the boundary should be detected and processed once the matrix $V$ or $X$ is updated. The specific method can be implemented by logical operations and Hadamard products. For a more visual description, IMPSO is illustrated with the matrix $X$ as an example, where $XB$ is the upper boundary, and the detection and processing of the upper boundary can be based on the following equation.

$$LOGIC_{N\times D} = X > (Ones \times XB) \tag{29}$$

where the $1 \times D$ matrix $XB$ is first expanded into an $N \times D$ matrix with each row equal to $XB$. Further, it is then compared with the $N \times D$ matrix $X$. If the elements of the matrix $X$ at the corresponding position are greater than the value of the upper boundary, the corresponding element position of the $N \times D$ matrix *LOGIC* is set to 1, and otherwise 0. With reference to this approach, the processing of the upper boundary can be implemented with the following equation.

$$X = LOGIC \circ XB + (1 - LOGIC) \circ X \tag{30}$$

The result of the operation is the element of matrix $X$ that is greater than the upper bound is set to the value of the upper bound. More specifically, the element of the matrix $X$ that is greater than the upper bound is set to 1 at the corresponding position in the matrix *LOGIC*, and thus the element of the matrix $X$ needs to be set to the value of the upper bound. Conversely, if an element of the matrix *LOGIC* is 0, it means that the element in the corresponding position of the matrix $X$ does not exceed the upper bound, then the element

of the matrix $X$ in the corresponding position of that element does not need to be changed either. The elements of the matrix $X$ that are smaller than the lower bound also need to be set to the value of the lower bound by a similar operation, which is not repeated here.

The next subsection describes in detail the two strategies used by the IMPSO algorithm to improve the population best fitness value, wherein the signal-to-noise distance is used to further update population best position on top of the basic update population position, and improved centroid opposition-based learning strategy is used to reinitialize population-related variables when the number of iterations is a multiple of 100 to exclude the influence of extreme values on the best fitness value, making the center of gravity of the population more representative.

### *3.2. Improved Opposition-Based Learning to Reinitialize the Population-Related Parameters*

Opposition-based learning is a computational intelligence scheme proposed by Tizhoosh [31] in 2005, which has been successfully applied to a variety of population-based evolutionary algorithms. Traditional learning strategies are essentially based on randomness, and once the worst-case scenario occurs, the search or optimization becomes unmanageable and the results take a lot of time to converge. The main idea of OBL is to consider both the points in the current space and their opposites and to select them meritedly with a view to obtaining results closer to the global optimum. In order to fully explore the current space and to make full use of the favorable information carried by the population as a merit-seeking whole, the COBL centroid opposition-based learning proposed by Rahnamayan et al. [32] was introduced on the basis of OBL.

**Theorem 1.** *The opposite point.*

*Suppose there exists a number x in* $[l, u]$*, then the opposite point of x is defined as*

$$x' = l + u - x \tag{31}$$

*Extending the definition of the opposite point to the D-dimension space, let* $p = (x_1, x_2, \ldots, x_D)$ *be a point in the D-dimension space, where* $x_i \in [l_i, u_i]$*,* $i = 1, 2, \ldots, D$*, then its opposite point is defined as*

$$p' = (x'_1,\ x'_2, \ldots, x'_D) \tag{32}$$

*where* $x'_i = l_i + u_i - x_i$*.*

**Theorem 2.** *Center of gravity.*

$(X1, \ldots, Xn)$ *is a group of n points with unit mass distributed in D-dimension space, and the center of gravity of the group is defined as*

$$M = \frac{(X1 + X2 + \ldots + Xn)}{n} \tag{33}$$

*It can also be expressed as.*

$$\frac{1}{n}\sum_{i=1}^{n} X_{i,j}, \quad j = 1, 2, \ldots, D \tag{34}$$

**Theorem 3.** *Center of gravity of the opposite point.*

*If the location of the center of gravity of a discrete uniform whole is M, then the opposite point of a point* $X_i$ *in the group is defined as*

$$X'_i = 2M - X_i,\ i = 1,\ 2,\ \ldots,\ n \tag{35}$$

*The opposite point is located in a search space with dynamic boundary, denoted* $X_{i,j} \in [a_j,\ b_j]$. *The dynamic boundary allows the search space to shrink continuously, which is calculated as*

$$a_j = min(X_{i,j}),\ b_j = max(X_{i,j}) \tag{36}$$

*where* $a_j$ *is the lower boundary of the search space, and* $b_j$ *is the upper boundary of the search space.*

*If the opposite point is outside the search boundary, the opposite point can be recalculated according to the following formula.*

$$\begin{cases} a_j + rand(0,1) \times (M_j - a_j), & if\ X_{i,j} < a_j \\ M_j + rand(0,1) \times (b_j - M_j), & if\ X_{i,j} > b_j \end{cases} \tag{37}$$

From the above, it is clear that the center-of-gravity position is chosen from the information of the average position of the population. In real life, people calculate the average value by removing the maximum and minimum values, so as to get rid of the influence of extreme values. In this paper, the center-of-gravity position is also calculated by subtracting the optimal position and the worst position to make the center-of-gravity position more representative. Using it for the initialization of the population will produce individuals that will be spread throughout the space, which is well prepared for the subsequent search for the best.

### *3.3. Signal-to-Noise Ratio Distance for Further Update the Position*

In the field of computer artificial intelligence, distance is a frequent and fundamental concept that has important applications in subfields such as natural language processing and computer vision. The concept of distance originates from the concepts of metrics and measurement in the field of mathematics. Distance is used in the computer field to represent the similarity between data; the greater the distance, the greater the degree of difference between the data. Common distance algorithms are Euclidean distance, Mahalanobis distance, Minkowski distance, etc. Among them, Euclidean distance is the most common representation of the distance between two or more points, but as the number of dimensions increases, the computation of the Euclidean distance increases substantially, which greatly increases the time overhead, and the difference between any two points in the space becomes weaker, leading to a uniform distribution of the data [33]. Hassanat et al. [34] uses the Euclidean norms and greedy algorithm to find the furthest pair of points (diameter) of a set of points in d-dimensional Euclidean feature space. On the other hand, the Euclidean distance treats the differences between the various dimensions of points in a space as equivalent, which sometimes does not satisfy the practical requirements. The Mahalanobis distance is a representation of the covariance distance of the data, and the Minkowski distance is a generalization of the Euclidean distance. In other words, the Minkowski distance can be expressed by a generalized formulation of several distance metric formulas, which can be degraded to Manhattan distance or Euclidean distance depending on the parameters, and the Chebyshev distance is the form in which the Minkowski distance takes its limit. Gueorguieva et al. [35] proposed an optimized fuzzy C-means clustering algorithm to improve the FCM clustering results by combining Mahalanobis distances and Minkowski distance metrics. Yang et al. [36] introduced signal-to-noise distance to measure the degree of difference between data, which can produce more discriminative features than the distance metric based on Euclidean distance [37], and the SNR distances of a pair of data $p_i$ and $p_j$ are defined as

$$d_S(p_i, p_j) = \frac{var(p_j - p_i)}{var(p_i)} = \frac{var(h_{ij})}{var(p_i)} \tag{38}$$

where $var(x) = \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}$ denotes the variance of $x$, $\mu = \frac{\sum_{i=1}^{n} xi}{n}$ denotes the mean of $x$, and $n$ denotes the dimension of $x$. The larger the SNR distance, the greater the degree of variance between the anchored and compared data.

Therefore, a new update mechanism that uses signal-to-noise ratio distance to determine the distance information between individuals and the optimal position was proposed in this paper. Through this distance, the worst position can be moved away from. The specific design formula is as follows.

$$d = var(x_i(t) - best(t))/var(best(t)) \tag{39}$$

$$x_i(t+1) = x_i(t) + sigmod(d)\cdot(x_i(t) - worst(t)) \tag{40}$$

In the formula, $x_i(t)$ denotes the position of the $i$-th individual in the $t$th generation, $best(t)$ denotes the best position in the $t$th generation, and $worst(t)$ denotes the worst position in the $t$th generation. It can be seen that $d$ determines the magnitude of individual search; the smaller $d$ is, the smaller the distance of individual $x_i(t)$ away from the worst position. On the contrary, the larger $d$ is, the larger the distance is. By adjusting individual position in this dynamic update, high-quality solutions can be searched adequately. The intelligence of the search is enhanced.

### *3.4. IMPSO Algorithm*

#### 3.4.1. IMPSO Algorithm Process

Input: The size of population *PopSize*, the dimension of the problem *PerLen*, the parameters $\omega$, $c1$, $c2$, maximal generation *max_iterations*.

Step 1. Initialize the matrices $X$ and $V$ according to Equations (19) and (20), control the elements of the matrix $X$ no greater than $XB$ and no less than $XM$; the elements of the matrix $V$ no greater than $VB$ and no less than $VM$.

Step 2. The fitness value of each individual of the matrix $X$, represented by the matrix *Fit*, is obtained from the Equation (21) in terms of individuals within the population.

Step 3. Update the best solution in terms of dimensions and select the individual with the best adaptation value for each dimension, i.e., each column, to form a matrix *gBest* of size $PopSize \times 1$.

Step 4. The best fitness value *gBest_Fit* is updated by the element with the best fitness value from the fitness value matrix *Fit*.

Step 5. Update the best position of an individual, specifically by using the matrix $X$ representing the position of the individual to obtain the personal best position matrix *pBest*.

Step 6. Update the matrix *pBest_Fit*, which represents the fitness values of the personal best positions with the matrix *Fit* representing the fitness values of all the individuals in the population.

Step 7. Perform *max_iterations* iterations for the following operations.

Step 8. Update velocity according to Equation (26).

Step 9. Using matrix $V$ as reference, if the element in matrix $V$ is greater than $VB$, set the element in the corresponding position in matrix *LOGIC* to 1; otherwise, set it to 0.

Step 10. Using the matrix *LOGIC*, the elements of the matrix $V$ greater than $VB$ are set to $VB$; otherwise, they remain unchanged.

Step 11. Using matrix $V$ as reference, if the element in matrix $V$ is smaller than $VM$, set the element in the corresponding position in matrix *LOGIC* to 1; otherwise, set it to 0.

Step 12. Using the matrix *LOGIC*, the elements of the matrix $V$ smaller than $VM$ are set to $VM$; otherwise, they remain unchanged.

Step 13. The personal position matrix $X$ is updated with the matrix $X$ and the latest obtained matrix $V$ according to Equation (27).

Step 14. Using matrix $X$ as reference, if the element in matrix $X$ is greater than $XB$, set the element in the corresponding position in matrix *LOGIC* to 1; otherwise, set it to 0.

Step 15. Using the matrix *LOGIC*, the elements of the matrix $X$ greater than $XB$ are set to $XB$; otherwise, they remain unchanged.

Step 16. Using matrix $X$ as reference, if the element in matrix $X$ is smaller than $XM$, set the element in the corresponding position in matrix $LOGIC$ to 1; otherwise, set it to 0.

Step 17. Using the matrix $LOGIC$, the elements of the matrix $X$ smaller than $XM$ are set to $XM$; otherwise, they remain unchanged.

Step 18. Update the matrix $Fit$ representing the fitness values of all the individuals with the latest obtained matrix $X$ according to Equation (21).

Step 19. Update the matrix $pBest$ and the matrix $pBest\_Fit$. If the matrix $pBest\_Fit$ is larger than the corresponding value in the matrix $Fit$, the corresponding element in the matrix $LOGIC$ is set to 1; otherwise, it is set to 0.

Step 20. If the matrix $pBest\_Fit$ is smaller than the corresponding value in the matrix $Fit$, it means that the updated personal position matrix is not as good as the previous personal position matrix, so the matrix $pBest$ that represents the personal best positions of all the individuals in the population does not need to be updated. Conversely, it means that the latest personal position matrix is better than the previous individual matrix, because the personal best fitness value is optimized, so it needs to be updated to the latest personal position matrix $X$.

Step 21. The matrix $Fit$ corresponds to the personal best fitness values of the population matrix $X$. The matrix $pBest\_Fit$ corresponds to the matrix $pBest$, and the best personal fitness values matrix is updated based on the personal best position matrix by comparing the previous equation.

Step 22. Using Equations (38)–(40) to further update the position of the population particles.

Step 23. Individuals with the best fitness values are selected in terms of dimensions, and the corresponding elements are assigned to the matrix $gBest$ according to the obtained individuals and dimensions in the matrix $pBest$.

Step 24. The element with the best fitness value is selected in the personal best fitness value matrix $pBest$, which is the best solution fitness value.

Step 25. When the number of iterations is a multiple of 100, the population-related variables are reinitialized using Equations (31)–(37). Exit the loop at the end of the iteration count; otherwise, go back to step8 to continue the iterations.

Output: The found best solution fitness $gBest\_Fit$.

The matrix $pBest$ represents the best personal positions of all the individuals in the IMPSO population. $pBest\_Fit$ is a matrix that selects the element with the best fitness value in all dimensions in terms of individuals, with a matrix size of $PopSize \times 1$. $gBest$ is a matrix that finds the corresponding row number of the best personal fitness value matrix $pBest\_Fit$, i.e., the individual with the best personal fitness value, in terms of dimensions, to achieve the goal of finding the individual with the best fitness value for each dimension, and the matrix size is $1 \times PerLen$. $gBest\_Fit$ is the matrix with the best fitness value in the personal best fitness value matrix $pBest\_Fit$.

#### 3.4.2. Flowchart Based on IMPSO Algorithm to Optimize DNA Sequence

To solve the problem of excessive time consumption and low quality in DNA sequence design optimization problems, this study proposes a multi-strategy matrix particle swarm and introduces an efficient matrix particle swarm to reduce the time consumption of the algorithm, then introduces novel centroid opposition-based learning to initialize the population during the optimization search to avoid the population falling into local states and finally introduces a signal-to-noise ratio to judge the distance between individuals for updates with high quality. The efficiency and reliability of DNA computing are inseparable from the design of the DNA chain. In order to design more excellent DNA sequences, it can be effective to combine the objective function and the constraints of the DNA chain. Before applying the objective function for calculation, the population particles are coded by dividing them by four, so that the matrix particle swarm can be coded with the four bases (A, C, G, T) of DNA. The specific algorithm flowchart is shown as Figure 1.

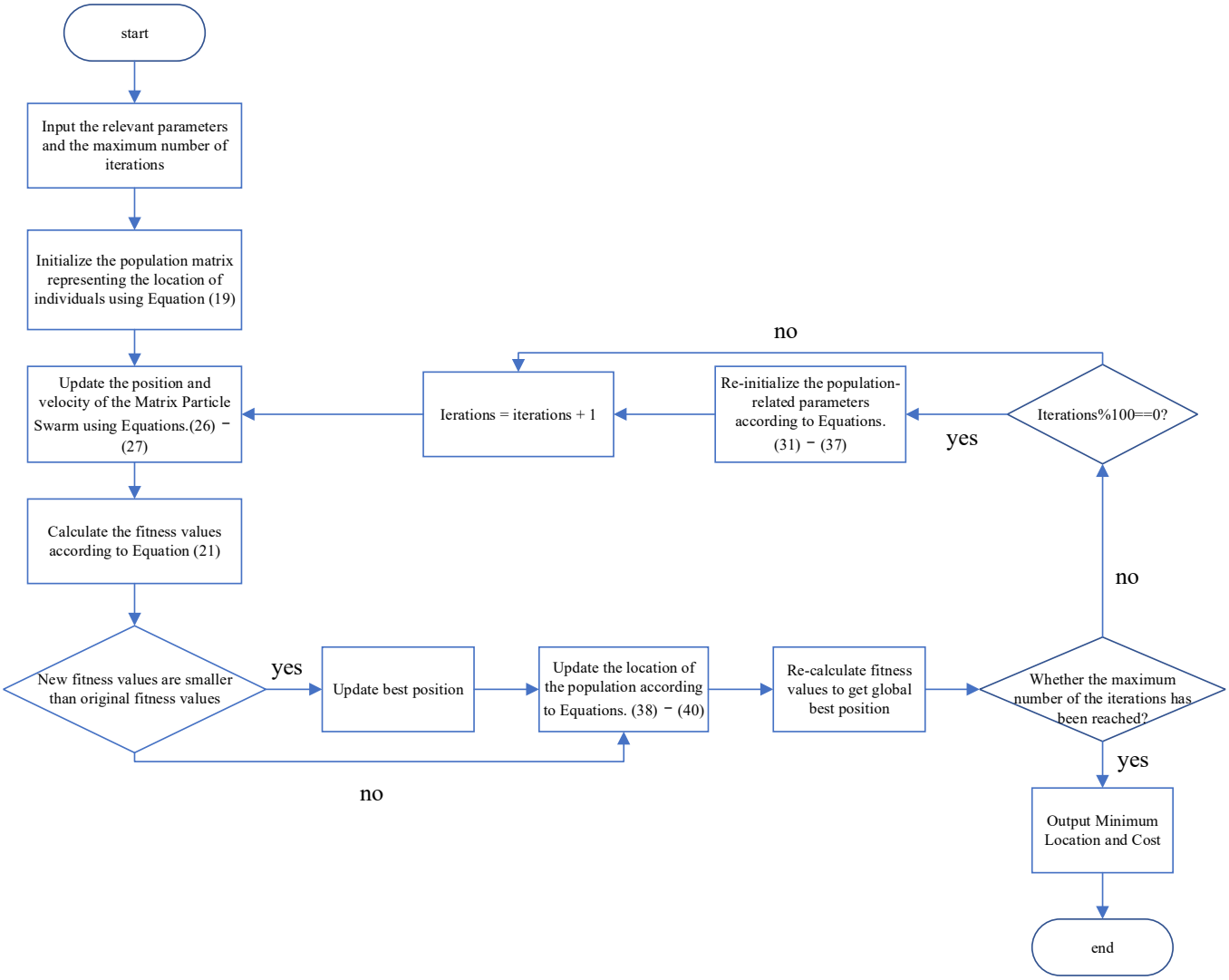


**Figure 1.** IMPSO algorithm flowchart.

## 4. Results and Analysis

### *4.1. Algorithm Parameters*

In this section, IMPSO is applied to DNA sequence design experimentation to demonstrate the high efficiency of the IMPSO in solving the DNA coding sequence design problem. All experiments were carried out on a computer with Intel (R) Core (TM) i5-10200H (2.40 Ghz) CPU, 16 GB RAM, 64-bit OS, and MATLAB R2020b simulation platform. In this experiment, the DNA molecule concentration is set to 10 nm, the salt solution concentration in the experimentation is set to 1 mol/L, the minimum values of the hair stem and hair loop were set to 6, and in the experiment on similarity and *H-Measure*, the penalty threshold for base continuity equality is set to 6, and, for discontinuity, it is set to 0.17. The continuity threshold for a single DNA strand is set to 2. The other parameters used in this study are described in Table 2.

### *4.2. Algorithm Results*

#### 4.2.1. Experimentation on the Effectiveness of IMPSO in Solving DNA Coding

To verify the feasibility of the IMPSO algorithm, the DNA sequences, the values of each constraint and their running times obtained from the optimization of IMPSO with MPSO, IWO, PSO and HS were compared. The results in Table 3 show that the IWO, PSO and HS algorithms take a long time to solve the DNA sequence design problem, all

above 20,000 s, and IWO even takes more than 35,000 s. The performance of MPSO shows that the running time of the swarm intelligence algorithm based on matrix operations is significantly reduced under the same conditions and that the values of each constraint of the DNA sequence do not become worse. The IMPSO algorithm requires more than two times more time compared to MPSO, which is due to the time required to add the improvement strategy. Although the time consumed increases, all the metrics of the DNA sequences obtained by IMPSO are better than those of MPSO, so the extra time consumption is worthwhile to obtain higher computational efficiency.

**Table 2.** Related parameters in IMPSO algorithm.

| Symbol | Implication | Value |
|---|---|---|
| Max_iteration | Maximum number of iterations | 3000 |
| PopSize | Size of the population | 20 |
| PerLen | Length of the individual | 20 |
| $XB$ | Upper bound | 3 |
| $XM$ | Lower bound | 0 |
| $VB$ | Maximum velocity constraint | 3 |
| $VM$ | Minimum velocity constraint | 0 |
| $\omega_{min}$ | Minimum number of dynamic constant | 0.4 |
| $\omega_{max}$ | Maximum number of dynamic constant | 0.9 |
| $C1_0$ | Initial factor for self-learning | 2.5 |
| $C1_{min}$ | Minimum factor for self-learning | 0.5 |
| $C2_0$ | Initial factor for social learning | 2.5 |
| $C2_{min}$ | Minimum factor for social learning | 0.5 |
| D | The size of Hamming Distance | 11 |

#### 4.2.2. Experimentations on the Competitiveness of IMPSO in Designing DNA Sequence

For demonstrating the competitiveness of IMPSO to solve DNA sequence design, this paper compares the experimentational DNA sequence design results of IMPSO with those of NCIWO, HSWOA, MO-ABC, CPSO and DMEA by comparing the average values of continuity, hairpin, *H-Measure*, similarity and the variance of Tm to assess sequence quality. Among these metrics, *H-Measure* and similarity are beneficial in preventing DNA strands from mismatching, and hairpin and continuity are beneficial in avoiding secondary structures in DNA strands. To ensure the fairness of the experimentations, parameters in the mentioned algorithm are set in accordance with their relevant references, and population size and iterations numbers were kept consistent.

### *4.3. Comparisons and Analysis*

Controlling continuity and hairpin structure in DNA sequences can prevent self-hybridization in DNA molecules to produce secondary structures and to ensure the reliability of DNA calculations. By constraining similarity and *H-Measure*, non-specific hybridization between a DNA sequence and its complementary sequences can be controlled. Melting temperature and free energy are important thermodynamic constraints of DNA molecules, and maintaining their stability is conducive to control the hybridization reaction between DNA molecules and to improve the reaction efficiency of DNA sequences.

#### 4.3.1. Control Secondary Structures

From the results in Table 4 and Figure 2, it can be seen that the continuity and hairpin of IMPSO and HSWOA are 0; however, the continuity or hairpin structures of NCIWO, MO-ABC, CPSO and DMEA exceed 0. This indicates that the DNA sequences created by IMPSO and HSWOA prevent secondary structures with advantage.

**Table 3.** Comparison of DNA sequences and their constraint values and Cputime.

| DNA Sequences (5′-3′) | Continuity | Hairpin | H-Measure | Similarity | Tm | GC% |
|---|---|---|---|---|---|---|
| IWO [23] | | | | | | |
| CCAACCTCCGAACCTACATA | 0 | 0 | 50 | 57 | 63.24 | 50 |
| CAGAACCAGAACAACGCCAA | 0 | 0 | 52 | 56 | 65.76 | 50 |
| ATTAACCACCTGCCTCTCTG | 0 | 0 | 54 | 54 | 63.85 | 50 |
| CGATTACACTCCTCACACCA | 0 | 0 | 51 | 56 | 63.78 | 50 |
| CAGCCAGGTGAAGATAAGAC | 0 | 0 | 59 | 53 | 62.33 | 50 |
| ACGGTGCTACCTGTTCCTAT | 0 | 0 | 61 | 54 | 65.13 | 50 |
| AGTATTGCGACGGCCTTCAA | 0 | 0 | 61 | 50 | 66.89 | 50 |
| Average | 0 | 0 | 55.43 | 54.29 | 64.42 | 50 |
| Cputime(s) | 35,379.59 | | | | | |
| PSO [24] | | | | | | |
| TACCTCCGTTCTTGCCACTT | 0 | 0 | 58 | 49 | 65.91 | 50 |
| CGGTGAGAGATGACGATTAG | 0 | 0 | 60 | 48 | 61.85 | 50 |
| ATAGCGTGACCAGCCAACAA | 0 | 0 | 63 | 49 | 66.88 | 50 |
| GTTGGATTGCGTACTCTCTG | 0 | 0 | 61 | 47 | 62.92 | 50 |
| TGTTGGTCAACCTGATGCTG | 0 | 0 | 64 | 49 | 65.25 | 50 |
| AGTTCTTAGGAGCGTGCAGA | 0 | 0 | 61 | 49 | 65.64 | 50 |
| CCGCCACACGAATCAATCTA | 0 | 0 | 63 | 47 | 64.81 | 50 |
| Average | 0 | 0 | 61.43 | 48.29 | 64.75 | 50 |
| Cputime(s) | 20,814.27 | | | | | |
| HS [25] | | | | | | |
| AGGAGAGACCTGGATTGAGT | 0 | 0 | 60 | 51 | 64.16 | 50 |
| TGTAGGAAGAGTGTGAACGG | 0 | 0 | 61 | 46 | 63.71 | 50 |
| GCAACCAACCATTACTCGAC | 0 | 0 | 57 | 50 | 63.78 | 50 |
| CCTTCCTTCCGCCTTATATC | 0 | 0 | 64 | 44 | 61.9 | 50 |
| AGGACATGAGAATCACACGG | 0 | 0 | 60 | 52 | 64.15 | 50 |
| GCAGAGACAATAACAAGCGG | 0 | 0 | 56 | 53 | 63.83 | 50 |
| GCCAATCAACATCGACACCT | 0 | 0 | 58 | 54 | 65.35 | 50 |
| Average | 0 | 0 | 59.43 | 50 | 63.84 | 50 |
| Cputime(s) | 21,364.64 | | | | | |
| MPSO [22] | | | | | | |
| TCCAAGCACACCATACCTCT | 0 | 0 | 58 | 50 | 65.39 | 50 |
| CGGAGAAGAAGTAGAACTGG | 0 | 0 | 55 | 51 | 61.66 | 50 |
| GACCACACTCAGGATCCATA | 0 | 0 | 58 | 55 | 62.96 | 50 |
| GCCAATATAGGCCACAGAGA | 0 | 0 | 64 | 50 | 63.69 | 50 |
| TCGCGTATCGTTGGTGTCTA | 0 | 0 | 65 | 48 | 65.66 | 50 |
| TTAACCGAGAATCTCGCAGG | 0 | 0 | 61 | 51 | 64.18 | 50 |
| ACATGAAGGTGCGGAAGCTT | 0 | 0 | 61 | 51 | 67.18 | 50 |
| Average | 0 | 0 | 60.29 | 50.86 | 64.39 | 50 |
| Cputime(s) | 6691.05 | | | | | |
| IMPSO | | | | | | |
| GGAGGTTAGGTTAGTGTTGG | 0 | 0 | 53 | 53 | 61.90 | 50 |
| CGACAAGAGATGAGAACACC | 0 | 0 | 54 | 49 | 62.57 | 50 |
| GAGTAGGTGAGATGGTAAGG | 0 | 0 | 47 | 55 | 60.80 | 50 |
| CAACGAACACGAACCAGTCA | 0 | 0 | 64 | 45 | 65.40 | 50 |
| GTTGGTGGTTGGTCCTTGTA | 0 | 0 | 58 | 47 | 64.57 | 50 |
| TATACCTAGAGTGAACGGCG | 0 | 0 | 61 | 50 | 63.04 | 50 |
| CCGCCATGAGGAAGTGTATA | 0 | 0 | 59 | 51 | 63.66 | 50 |
| Average | 0 | 0 | 56.57 | 50 | 63.13 | 50 |
| Cputime(s) | 15,008.97 | | | | | |

**Table 4.** Comparison of DNA sequences and corresponding constraint values.

| DNA Sequences (5′-3′) | Continuity | Hairpin | H-Measure | Similarity | Tm | GC% |
|---|---|---|---|---|---|---|
| IMPSO | | | | | | |
| GGAGGTTAGGTTAGTGTTGG | 0 | 0 | 53 | 53 | 61.90 | 50 |
| CGACAAGAGATGAGAACACC | 0 | 0 | 54 | 49 | 62.57 | 50 |
| GAGTAGGTGAGATGGTAAGG | 0 | 0 | 47 | 55 | 60.80 | 50 |
| CAACGAACACGAACCAGTCA | 0 | 0 | 64 | 45 | 65.40 | 50 |
| GTTGGTGGTTGGTCCTTGTA | 0 | 0 | 58 | 47 | 64.57 | 50 |
| TATACCTAGAGTGAACGGCG | 0 | 0 | 61 | 50 | 63.04 | 50 |
| CCGCCATGAGGAAGTGTATA | 0 | 0 | 59 | 51 | 63.66 | 50 |
| Average | 0 | 0 | 56.57 | 50 | 63.13 | 50 |
| HSWOA [26] | | | | | | |
| CTCGTCTAACCTTCTTCAGC | 0 | 0 | 63 | 51 | 62.28 | 50 |
| CTGTGTGGAATGCAAGGATG | 0 | 0 | 64 | 48 | 63.82 | 50 |
| CGAGCGTAGTGTAGTCATCA | 0 | 0 | 63 | 69 | 63.56 | 50 |
| AGTTACAGGACACCACCGAT | 0 | 0 | 65 | 51 | 66.39 | 50 |
| CAGTAGCAGTCATAACGAGC | 0 | 0 | 64 | 56 | 62.69 | 50 |
| GCATAGCACATCGTAGCGTA | 0 | 0 | 59 | 54 | 64.60 | 50 |
| TGGACCTTGAGAGTGGAGAT | 0 | 0 | 62 | 50 | 64.44 | 50 |
| Average | 0 | 0 | 62.86 | 54.14 | 63.97 | 50 |
| NCIWO [9] | | | | | | |
| ACACCAGCACACAGAAACA | 9 | 0 | 55 | 46 | 66.99 | 50 |
| GTTCAATCGCCTCTCGGTAT | 0 | 0 | 57 | 52 | 64.26 | 50 |
| GCTACCTCTTCCACCATTCT | 0 | 0 | 55 | 53 | 63.55 | 50 |
| GAATCAATGGCGGTCAGAAG | 0 | 0 | 66 | 47 | 63.58 | 50 |
| TTGGTCCGGTTATTCCTTCG | 0 | 0 | 65 | 52 | 64.44 | 50 |
| CCATCTTCCGTACTTCACTG | 0 | 0 | 56 | 56 | 62.30 | 50 |
| TTCGACTCGGTTCCTTGCTA | 0 | 0 | 58 | 54 | 65.61 | 50 |
| Average | 1.29 | 0 | 58.86 | 51.43 | 64.39 | 50 |
| MO-ABC [8] | | | | | | |
| GTAAGGAAGGCAAGGCAGAA | 0 | 0 | 42 | 54 | 64.70 | 50 |
| GTTGGTGGTTGTTGGTGGTT | 0 | 0 | 46 | 36 | 66.00 | 50 |
| GGAGACGGAATGGAAGAGTA | 0 | 0 | 44 | 55 | 62.93 | 50 |
| CCATTCTTCTCTTCTCTCCC | 9 | 0 | 67 | 22 | 61.39 | 50 |
| AGGAGAGGAGAGGAGGAAAA | 16 | 0 | 31 | 53 | 63.80 | 50 |
| ATAAGAGAGAGAGAGAGGGG | 16 | 0 | 34 | 51 | 61.11 | 50 |
| GAGCCAACAGCCAACCAAAA | 16 | 0 | 48 | 45 | 66.40 | 50 |
| Average | 8.14 | 0 | 44.57 | 45.14 | 63.76 | 50 |
| CPSO [27] | | | | | | |
| GACCGGTAAGATGAAGAGGA | 0 | 0 | 60 | 50 | 62.94 | 50 |
| CTATGCTTCTATCGCCTTCC | 0 | 0 | 61 | 51 | 62.23 | 50 |
| TAGTTGCACGAGAGAAGCAG | 0 | 0 | 60 | 51 | 64.38 | 50 |
| CGTGTACGAGCCTAATAAGG | 0 | 0 | 64 | 54 | 62.14 | 50 |
| CTTTGTCCATTGCACATCCG | 9 | 0 | 61 | 53 | 64.42 | 50 |
| TCCTATCCGAGATGATCCGT | 0 | 3 | 63 | 55 | 64.08 | 50 |
| TTCAACTTACGCTGTACGGC | 0 | 6 | 63 | 54 | 65.25 | 50 |
| Average | 1.29 | 1.29 | 61.71 | 52.57 | 63.63 | 50 |
| DMEA [28] | | | | | | |
| TGAGTTGGAACTTGGCGGAA | 0 | 0 | 70 | 52 | 66.76 | 50 |
| CAGCATGTTAGCCAGTACGA | 0 | 0 | 60 | 55 | 64.65 | 50 |
| TTGAGTCCGCGTGGTTGGTC | 0 | 0 | 63 | 53 | 69.79 | 60 |
| AATTGACACTCTGATTCCGC | 0 | 0 | 73 | 58 | 62.89 | 45 |
| CATACATTGCATCAACGGCG | 0 | 0 | 67 | 53 | 64.84 | 50 |
| ATACACGCACCTAGCCACAC | 0 | 0 | 59 | 50 | 66.93 | 55 |
| GTTCCACAACAGGTCTAATG | 0 | 3 | 61 | 53 | 60.65 | 45 |
| Average | 0 | 0.43 | 64.71 | 53.43 | 65.22 | 50.71 |

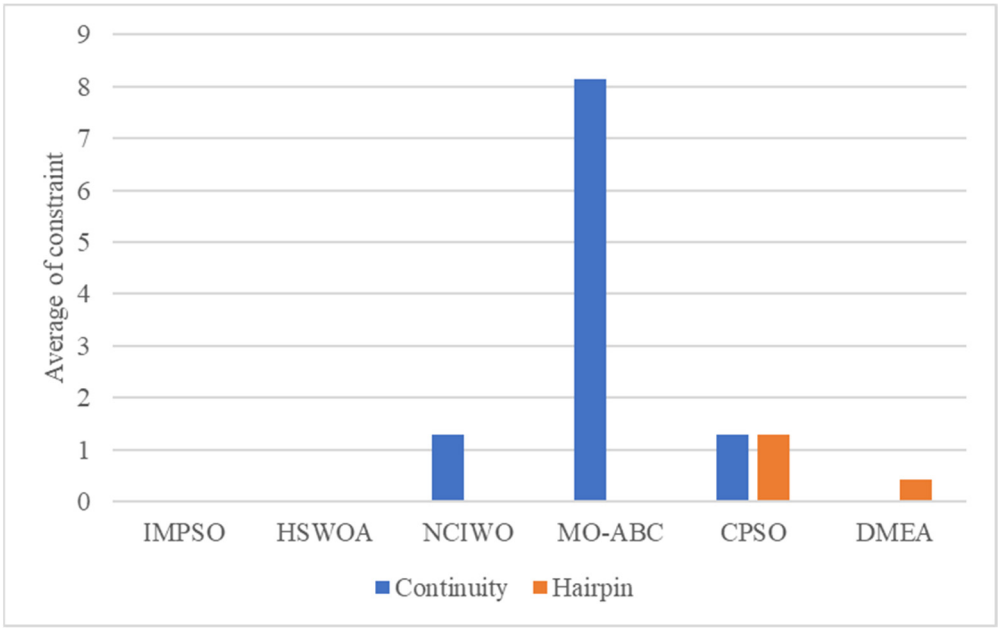


**Figure 2.** Comparison results among average values of IMPSO, HSWOA, NCIWO, MO-ABC, CPSO, DMEA and IMPSO in continuity and hairpin.

#### 4.3.2. Control Nonspecific Hybridization

From Table 4 and Figure 3, *H-Measure* and similarity values of IMPSO are more desirable than other algorithms, only second to MO-ABC, due to their priority to the constraints set of *H-Measure* and similarity at the expense of continuity and hairpin structure, so the sequences of IMPSO are overall superior to those of MO-ABC.

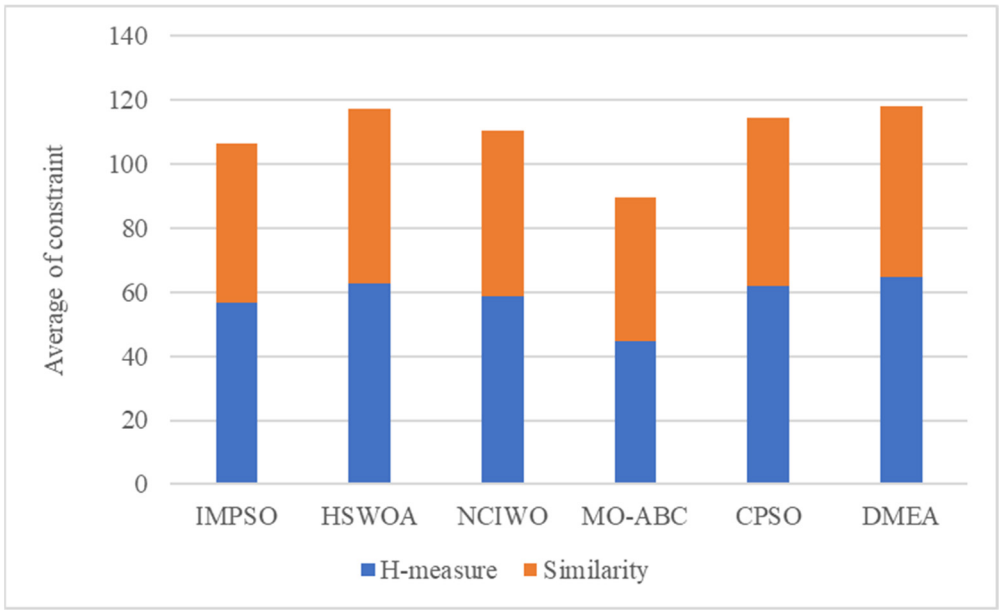


**Figure 3.** Comparison results among average values of HSWOA, NCIWO, MO-ABC, CPSO, DEMA and IMPSO in *H-Measure* and similarity.

#### 4.3.3. Thermodynamics of Tm

In DNA calculation, DNA sequences need to be as consistent as possible in terms of Tm to dominate biochemical reactions. In this experiment, the variance was used to measure the fluctuation of the Tm of the DNA sequences generated by each algorithm.

From Table 4 and Figure 4, the variance of Tm of IMPSO is superior to MO-ABC and DMEA and slightly inferior to CPSO, HSWOA and NCIWO.

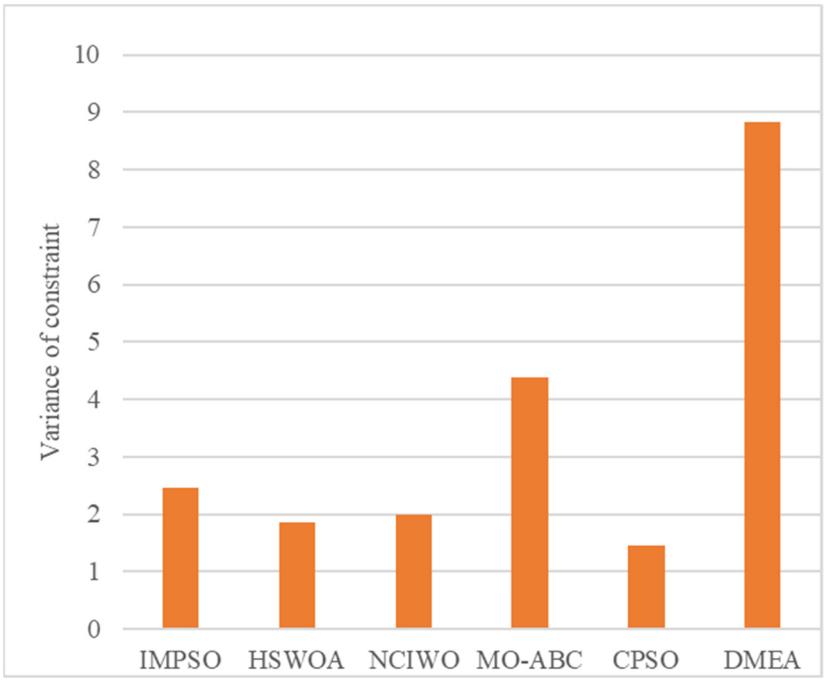


**Figure 4.** Comparison results among average values of HSWOA, NCIWO, MO-ABC, CPSO, DEMA and IMPSO in Tm variance.

## 5. Conclusions

To preferably solving the problem of DNA sequence optimization design, an improved multi-strategy matrix particle swarm optimization algorithm is proposed in this paper, which uses an approach in accordance with the signal-to-noise ratio distance to dynamically update the optimal and worst positions of individuals within the population and can adequately search for high-quality solutions. The centroid opposition-based learning strategy is introduced to improve the search range of the algorithm and to exclude the extreme differences brought by the optimal and worst positions when calculating the center-of-gravity positions, so that the center-of-gravity positions are more representative. The individuals generated in the initialization of the population of matrix particles can be spread over the whole space, making full use of the favorable information carried by the population as a whole in the search for the global best, avoiding the premature convergence of the population into a local optimum and fully preparing for the subsequent search for the global optimum. Finally, matrix operations are used to greatly reduce the algorithm running time and to obtain higher computational efficiency without sacrificing the DNA constraint values. Experiments comparing with other particle swarm algorithms confirm that, excluding the MPSO algorithm, the runtime of the swarm intelligence algorithm based on matrix operations is significantly reduced under the same conditions, that various constraint values of DNA sequences do not become worse compared with other algorithms and that the comprehensive capability and reliability of DNA computation are outstanding. The improved multi-strategy matrix particle swarm algorithm (IMPSO) does not underperform in terms of DNA constraint values compared with other DNA sequence design experiments, taking into account the global picture and obtaining optimized sequences of high quality, verifying the effectiveness of the algorithm and meeting the requirements for application to DNA computation. However, the individual capabilities under the combined capability, especially the melting temperature variance, need to be improved. By not sacrificing the DNA constraint values and making full use of the whole population diversity, the CPU running time will also be increased. How to find a breakthrough point to gradually improve

the single-item capability without sacrificing any necessary constraint to achieve a more excellent DNA computation capability is also something that needs further consideration in future work.

**Author Contributions:** Data curation, W.Z.; formal analysis, W.Z. and Z.H.; funding acquisition, C.Z.; software, W.Z. and D.Z.; supervision, D.Z.; validation, C.Z. and Z.H.; writing—review and editing, W.Z. and D.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by the National Natural Science Foundation of China under Grant numbers 62272418, and 62002046.

**Data Availability Statement:** Dataset used in this study may be available on demand.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Watson, J.D.; Crick, F.H. Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. *Nature* **1953**, *171*, 737–738. [CrossRef]
2. Adleman, L.M. Molecular Computation of Solutions to Combinatorial Problems. *Science* **1994**, *266*, 1021–1024. [CrossRef] [PubMed]
3. Seelig, G.; Soloveichik, D.; Zhang, D.Y.; Winfree, E. Enzyme-free Nucleic Acid Logic Circuits. *Science* **2006**, *314*, 1585–1588. [CrossRef] [PubMed]
4. Church, G.M.; Gao, Y.; Kosuri, S. Next-generation Digital Information Storage in DNA. *Science* **2012**, *337*, 1628. [CrossRef] [PubMed]
5. Extance, A. How DNA Could Store All the World's Data. *Nature* **2016**, *537*, 22–24. [CrossRef]
6. Zhirnov, V.; Zadegan, R.M.; Sandhu, G.S.; Church, G.M.; Hughes, W.L. Nucleic Acid Memory. *Nat. Mater.* **2016**, *15*, 366–370. [CrossRef]
7. Zhu, D.L.; Huang, Z.W.; Liao, S.G.; Zhou, C.J.; Yan, S.Q.; Chen, G. Improved Bare Bones Particle Swarm Optimization for DNA Sequence Design. *IEEE Trans. NanoBioscience* **2022**. [CrossRef]
8. Chaves-González, J.M.; Vega-Rodríguez, M.A.; Granado-Criado, J.M. Multiobjective Swarm Intelligence Approach Based on Artificial Bee Colony for Reliable DNA Sequence Design. *Eng. Appl. Artif. Intell.* **2013**, *26*, 2045–2057. [CrossRef]
9. Yang, G.J.; Wang, B.; Zheng, X.; Zhou, C.J.; Zhang, Q. IWO Algorithm Based on Niche Crowding for DNA Sequence Design. *Interdiscip. Sci. Comput. Life Sci.* **2017**, *9*, 341–349. [CrossRef]
10. Zhang, K.; Xu, J.; Geng, X.T.; Xiao, J.H.; Pan, L.Q. Improved Taboo Search Algorithm for Designing DNA Sequences. *Prog. Nat. Sci.* **2008**, *18*, 623–627. [CrossRef]
11. Cervantes-Salido, V.M.; Jaime, O.; Brizuela, C.A.; Martínez-Pérez, I.M. Improving the Design of Sequences for DNA Computing: A Multiobjective Evolutionary Approach. *Appl. Soft Comput.* **2013**, *13*, 4594–4607. [CrossRef]
12. Chaves-González, J.M.; Vega-Rodríguez, M.A. DNA Strand Generation for DNA Computing by Using A Multi-objective Differential Evolution Algorithm. *Biosystems* **2014**, *116*, 49–64. [CrossRef]
13. Chaves-González, J.M.; Vega-Rodríguez, M.A. A Multiobjective Approach Based on The Behavior of Fireflies to Generate Reliable DNA Sequences for Molecular Computing. *Appl. Math. Comput.* **2014**, *227*, 291–308. [CrossRef]
14. Eberhart, R.; Kennedy, J. A New Optimizer Using Particle Swarm Theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43. [CrossRef]
15. Houssein, E.H.; Gad, A.G.; Hussain, K.; Suganthan, P.N. Major Advances in Particle Swarm Optimization: Theory, Analysis, and Application. *Swarm Evol. Comput.* **2021**, *63*, 100868. [CrossRef]
16. Ghatasheh, N.; Faris, H.; Abukhurma, R.; Castillo, P.A.; Al-Madi, N.; Mora, A.M.; Al-Zoubi, A.M.; Hassanat, A. Cost-sensitive Ensemble Methods for Bankruptcy Prediction in A Highly Imbalanced Data Distribution: A Real Case from the Spanish Market. *Prog. Artif. Intell.* **2020**, *9*, 361–375. [CrossRef]
17. Zhang, Q.K.; Liu, W.G.; Meng, X.X.; Yang, B.; Vasilakos, A.V. Vector coevolving particle swarm optimization algorithm. *Inf. Sci.* **2017**, *394*, 273–298. [CrossRef]
18. Coello, C.A.C.; Lechuga, M.S. MOPSO: A Proposal for multiple objective particle swarm optimization. In Proceedings of the 2002 Congress on Evolutionary Computation Part of the 2002 IEEE World Congress on Computational Intelligence, Honolulu, HI, USA, 12–17 May 2002; Volume 2, pp. 1051–1056. [CrossRef]
19. Corne, D.W.; Jerram, N.R.; Knowles, J.D.; Oates, M.J. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In Proceedings of the 3rd Annual Conference on Genetic And Evolutionary Computing Conference, San Francisco, CA, USA, 7–11 July 2001; pp. 283–290. [CrossRef]
20. Hu, X.H.; Eberhart, R. Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization. In Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; pp. 1677–1681. [CrossRef]
21. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.A.M.T. A Fast And Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2022**, *6*, 182–197. [CrossRef]

22. Zhan, Z.H.; Zhang, J.; Lin, Y.; Li, J.Y.; Huang, T.; Guo, X.Q.; Wei, F.F.; Kuang, S.X.; Zhang, X.Y.; You, R. Matrix-Based Evolutionary Computation. *IEEE Trans. Emerg. Top. Comput. Intell.* **2021**, *6*, 315–328. [CrossRef]
23. Mehrabian, A.R.; Lucas, C. A Novel Numerical Optimization Algorithm Inspired from Weed Colonization. *Ecol. Inform.* **2006**, *1*, 355–366. [CrossRef]
24. Poli, R.; Kennedy, J.; Blackwell, T. Particle Swarm Optimization. *Swarm Intell.* **2007**, *1*, 33–57. [CrossRef]
25. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A New Heuristic Optimization Algorithm: Harmony Search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
26. Xue, L.; Wang, B.; Lv, H.; Yin, Q.; Zhang, Q.; Wei, X.P. Constraining DNA Sequences with A Triplet-bases Unpaired. *IEEE Trans. NanoBiosci.* **2020**, *19*, 299–307. [CrossRef]
27. Liu, Y.Y.; Zheng, X.D.; Wang, B.; Zhou, S.H. The Optimization of DNA Encoding Based on Chaotic Optimization Particle Swarm Algorithm. *J. Comput. Theor. Nanosci.* **2016**, *13*, 443–449. [CrossRef]
28. Xiao, J.H.; Jiang, Y.; He, J.J.; Cheng, Z. A Dynamic Membrane Evolutionary Algorithm for Solving DNA Sequences Design with Minimum Free Energy. *MATCH Commun. Math. Comput. Chem.* **2013**, *70*, 971–986.
29. Shin, S.Y.; Lee, I.H.; Kim, D.; Zhang, B.T. Multiobjective Evolutionary Optimization of DNA Sequences for Reliable DNA Computing. *IEEE Trans. Evol. Comput.* **2005**, *9*, 143–158. [CrossRef]
30. Watkins, N.E., Jr.; SantaLucia, J., Jr. Nearest-neighbor Thermodynamics of Deoxyinosine Pairs in DNA Duplexes. *Nucleic Acids Res.* **2005**, *33*, 6258–6267. [CrossRef] [PubMed]
31. Tizhoosh, H.R. Opposition-based Learning: A New Scheme for Machine Intelligence. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vienna, Austria, 28–30 November 2005; Volume 1, pp. 695–701. [CrossRef]
32. Rahnamayan, S.; Jesuthasan, J.; Bourennani, F.; Salehinejad, H.; Naterer, G.F. Computing Opposition by Involving Entire Population. In Proceedings of the IEEE Congress on Evolutionary Computation, Beijing, China, 6–11 July 2014; pp. 1800–1807. [CrossRef]
33. Milman, V.D. New Proof of the Theorem of A. Dvoretzky on Intersections of Convex Bodies. *Funct. Anal. Its Appl.* **1971**, *5*, 288–295. [CrossRef]
34. Hassanat, A.B.A. Furthest-Pair-Based Decision Trees: Experimentational Results on Big Data Classification. *Information* **2018**, *9*, 284. [CrossRef]
35. Gueorguieva, N.; Valova, I.; Georgiev, G. M&MFCM: Fuzzy C-means Clustering with Mahalanobis and Minkowski Distance Metrics. *Procedia Comput. Sci.* **2017**, *114*, 224–233. [CrossRef]
36. Yang, J.H.; Yu, J.H.; Huang, C. Adaptive Multistrategy Ensemble Particle Swarm Optimization with Signal-to-Noise Ratio Distance Metric. *Inf. Sci.* **2022**, *612*, 1066–1094. [CrossRef]
37. Yuan, T.T.; Deng, W.H.; Tang, J.; Tang, Y.N.; Chen, B.H. Signal-To-Noise Ratio: A Robust Distance Metric for Deep Metric Learning. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 4810–4819. [CrossRef]