RESEARCH ARTICLE

Prediction of RNA Secondary Structure Using Butterfly Optimization Algorithm

Sajib Chatterjee[1] · Rameswar Debnath[1] · Sujit Biswas[2] · Anupam Kumar Bairagi[1]

Abstract

Ribonucleic acid (RNA) structure is vital to its ability to function within the cell. The ability to predict RNA structure is essential to implementing new medications and understanding genetic illnesses. It is also important in synthetic and computational biology. All these functions are directly related to its secondary structure. Also prediction of RNA secondary structure process is the most significant step to determining the tertiary structure of RNA. On account of this, prediction of secondary structure of RNA is the crying topic in bioinformatics. In this research, we present the swarm-based metaheuristic Butterfly Optimization Algorithm (BOA) method for predicting the secondary structure of RNA. The main feather of the BOA is that it can conduct both local and global search simultaneously. According to the problem perspective, we have redesigned the operators of BOA to perform global and local search operations in different ways. We have followed a thermodynamic model for the selection of the stable secondary structure with minimum Gibbs free energy. Predicting the minimum free energy value we also developed an "Optimize" function to search the new optimize structure. This function increases the prediction efficiency, creating new stable structure and also decreases the time complexity of global searching procedure. We have used a public dataset to perform the prediction operation. To accuse our prediction efficiency, we have compared our outcomes to existing popular algorithms. The result shows that the proposed approach can predict secondary RNA structure better than other state-of-the-art algorithms.

Keywords RNA structure · Secondary structure prediction · Butterfly optimization algorithm · Fragrance · Optimize function

1 Introduction

RNA is one biological macromolecules that is most significant for all known forms of life and has a similar structure to DNA but differs in minor ways. RNA holds significant importance in protein synthesis, transferring genetic information into the cells, DNA replication, managing gene activity during evolution, cellular differentiation, and participating in genetic evolution. Protein synthesis involves three different varieties of RNA, namely ribosomal RNA (rRNA), transfer RNA (tRNA), and messenger RNA (mRNA). Where mRNA contains information for protein synthesis, tRNA transports amino acids to ribosomes as input for protein synthesis and rRNA is the core of a cell's ribosomes [1]. RNA is a molecules consisting of long chain of nucleotides, and nucleotides in RNA are classified into four types: adenine (A), uracil (U), guanine (G), and cytosine (C).

In RNA, there are three types of structure. The primary structure is the strings of nucleotide in a straight-line-sequence like GCCUCAUGGUGGUGGCUGGGGGCA GCCUCAUGGUGGU GGUGGCUGGGG. This structure serves as a means of distinguishing one RNA from another and notifies short information about the RNA structure. The secondary structure (2D bonded base pair) is a folding of the molecule on itself by forming hydrogen bonds between C-G, A-U and G-U. The complimentary nucleotides are connected by hydrogen bonds. Whereas (C-G) and (A-U) are regular canonical base pairs, also called the Watson–Crick base pair, and (G-U) is the less stable non-canonical 'Wobble' base pair [2]. Forming canonical and non-canonical base pairs in RNA secondary structure involves the folding operation [3].

✉ Anupam Kumar Bairagi
anupam@cse.ku.ac.bd

1 Computer Science and Engineering Discipline, Khulna University, Khulna 9208, Bangladesh

2 Department of Computer Science and Digital Technology, University of East London, London, UK

Hence, predicting of the RNA secondary structure returns to predict all the hydrogen bonds from the primary structure of the sequence. Tertiary structure provides a three-dimensional view of RNA molecules.

Predicting the secondary structure is extensively viewed as the first step towards recognizing the function of an RNA molecule. Researchers have been focusing on determining the secondary structure of RNA for several decades because understanding hereditary illnesses and discovering new treatments are the most pressing concerns. It also aids biologists in determining the importance of a substance in a cell [4]. The structure of RNA aids in understanding the functionality of RNA. Physical methods like Nuclear Magnetic Resonance (NMR) and X-ray crystallography have been developed to anticipate the structure of RNA. However, these procedures are complex, time-consuming, and costly [5]. Researchers have recently concentrated on applying mathematical and computational tools to determine the best strategy to address RNA structural difficulties. Many approaches and algorithms [6–11] have recently been developed to handle RNA secondary structure difficulties.

The RNA secondary structure prediction problem is predicting from a primary RNA sequence its secondary structure representation. The problem is declared to be NP-hard. The most important factor on predicting accuracy of RNA structure is the length of the sequence. Usually with the increasing of the molecule size in an RNA sequence the accuracy gets low [12]. A dynamic programming approach based on free energy minimization with a polynomial complexity of $O(n^6)$. Using it in practice, especially for long-length RNA sequences, poses a significant challenge [13]. Utilizing dynamic programming with a focus on minimizing free energy reveals that thermodynamic models employed for estimating the free energy of an RNA secondary structure are generally accurate only within a 5–10% margin. This is problematic since many RNA secondary structures lie within 5–10% of the global minimum free energy. Another hurdle in this issue is that heuristic approaches offer no assurance of locating the structure with the minimum free energy, yet they can be faster and more adept at handling long length RNA sequences. Furthermore, they are inherently far less constraining compared to dynamic programming algorithms concerning the complexity of the underlying energy model. For that reason it is not necessary to find the global minimum free energy. In this scenario it's good enough to get most stable structure with close to the minimum free energy.

In this paper, we present a naturally inspired swarm-based metaheuristic Butterfly Optimization Algorithm (BOA) to investigate the RNA secondary structure. To resolve this issue, we have followed the food-searching nature of butterflies and performed local and global search operations premised on the sensitivity of butterfly fragrance. These operations help us find a stable structure and optimum solution. To predict RNA secondary structure, we have followed some steps. We have designed four operators separate global search, reverse global search, exchange local search and marge local search. Separate global search divides each molecules or structure into two structures inject random elements that help to find global minima point. Reverse global search combine two different region local minima points to explore to search global minimum structure. Exchange local search exchanges different monomer position in a local region of the search space and works like mutation operator for create little changes among the molecules. Marge local search chose two different structures in a local region area and marge the even and odd position between this two structures to create local search minima points. An "Optimize" function discard extra duplicate point among the structure and that's speed up the searching procedure. The advanced optimize function optimizes the search operation result and decrease the time complexity of the process. In recent years, BOA has been used to solve many optimizations problem, and this algorithm gives a better result than existing algorithms for classification and optimization problems. Such as feature selection problem [14], Node Localization in Wireless Sensor Networks problem [15], A Self-Adaption Numerical Optimization Problem [16], Evolving Artificial Neural Networks Data Classification problem [17], a novel approach for global optimization problem [18] and protein folding optimization problem [19]. For the robust performance of BOA algorithm for optimization problem we have chosen BOA for solving RNA secondary structure prediction problem.

The major contributions and novelty of this work are summarized as follow:

- A novel efficient approach has been developed using Butterfly Optimization Algorithm (BOA). BOA applied for different NP-hard problem but not used for RNA secondary structure prediction problem.
- We have designed both local and global search with four operators: separate, reverse, exchange and marge. All these operators make the searching process of best structure of RNA sequence robust and convenient.
- Optimize butterfly function is another novel task. We implement this function to remove extra duplicate stem number(s). If the four operators of BOA produced duplicate stem number(s) then optimize butterfly discard the duplicate and make valid structure. This operator makes the BOA most time efficient.

The paper layout is as follow: the related works of RNA structure prediction will be described in Sect. 2. The butterfly optimization algorithm based on RNA structure prediction problem has been illustrate and describe in Sect. 3. The experimental results and comparison with the state of the art

paper of RNA secondary structure prediction problem are shown in Sect. 4. The conclusion of the paper is described in Sect. 5.

2 Related Works

To anticipate the RNA secondary structure, many approaches or algorithms have been devised. Dynamic programming, heuristics and metaheuristic algorithms are applied for solving this NP-hard problem. The efficient algorithms are reviewed below.

2.1 Dynamic Programming (DP)

Dynamic programming (DP) is a technique for decomposing any big problem into smaller ones in order to solve it. Each tiny problem is then resolved, the outcomes are stored, and the recursive method is used to calculate the results afterward. Tatsuya Akutsu [13] proposed Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. The time complexity of the method increases to $O(n^5)$ or more when complicated pseudoknots are handled. Another important difficulty is that this method has no identification of what class of pseudoknot it should cover and no established energy function specifically for the loop regions. Zuker [20] created the DP-based method m-fold for creating the RNA secondary structure with the least amount of free energy possible. The dynamic programming algorithm cannot consider the kinetic factors related to easily accessible states in RNA folding. Kengo Sato and Yuki Kato [21] developed a linear partition model that enhances the prediction of secondary structures in long sequences by considering pseudoknots. Despite this advancement, the model's prediction accuracy for crossing base pairs remains suboptimal.

2.2 Deep Learning

In their work presented in [22], the authors introduced an algorithm based on deep learning, aimed at predicting RNA secondary structures. This approach integrates thermodynamic principles into its framework. The effectiveness of machine learning-based methods, such as this one, is anticipated to enhance prediction accuracy as the volume of training data increases. Additionally, the authors identified that a significant challenge in predicting secondary structures from single sequences is the absence of prior knowledge about the sequences involved. In paper [23], the researchers introduced REDfold, an innovative algorithm for predicting RNA secondary structure. This algorithm employs a residual encoder-decoder learning network as its core. Unlike traditional methods that rely on dynamic programming, REDfold

uses constrained optimization. This approach allows for the prediction of structures beyond the limitations of nested folding patterns. Fu et al. [24] proposed UFold: fast and accurate RNA secondary structure prediction with deep learning. Without accessing more training data, achieving higher accuracy in predictions is not possible here.

2.3 Heuristic Algorithms

By the nature of various bioinformatics problems are very difficult to solve optimally and inside the polynomial time of their size. For that reason, bioinformatics motivates to the use of heuristic algorithms. A heuristic algorithm represent an algorithmic structure that are enable to produce an adequate solution of a problem within polynomial time in the real scenarios, but the solution is the actual optimal result there is no formal proof of its. Under the given constraints of a problem, when there is no familiar method to find an optimal solution then heuristic algorithms are typically used.

2.3.1 Genetic Algorithm (GA)

Genetic algorithm (GA) and simulated annealing (SA) are used for predicting RNA secondary structure prediction a hybrid framework approach in [25]. The authors created combination of these two algorithms. For the combination of GA and SA, GA is applied for a global search and SA is applied for a local search, and moreover for the combination of SA and GA, where SA is applied for a global search and GA is applied for a local search. The hybridization of GA and SA provide better perform than the single GA and SA. But the main pitfall of the algorithm it can predict only 2 order pseudoknots and for complex pseudoknots it could not predict accurate structure.

Based on thermodynamic models, Wiese developed the RNA predict technique [10]. Specifically, they assessed the causative link between the number of true positive base pairs and the free energy in a structure in that paper's first section to evaluate the effectiveness of the Individual Nearest Neighbor Hydrogen Bond (INN-HB) and the grouping energy-based thermodynamic models. Tong and Cheung [26] suggested a different strategy GAknot. Using GA they predicted pseudoknots RNA secondary structures. In addition to making it likely to look for MFE structures, GA provides several solutions that are substandard structures and other structures that are more closely related to the normal fold. While GA is useful for determining basic energy parameters but it's time-consuming for many helices.

2.3.2 Tabu Search (TS)

In [27], the authors proposed a tabu search-based RNA secondary structure prediction model RNATS. They use two

different search models: intensification and diversification. These search models use the immediate regions surrounding the existing solution, explore the previously unexplored territory, and execute prediction operations using the minimal free energy technique. Then they experimentally analyzed their proposed method with six RNA sequences and the outcome gave a compelling performance.

2.3.3 Simulated Annealing (SA)

Schmitz and Seger [28] were the first to suggest utilizing Simulated Annealing to identify RNA secondary structures using a free energy optimization technique where the Iterative model and breakdown of single base pairs determine the secondary structure. Tsang and Grypma developed a permutation-based RNA Structure prediction method [3]. In this paper, they have applied swap mutation operator with adaptive annealing scheduling and flip mutation operator for geometric scheduled simulated annealing. Here, only sequences with lower energies can benefit from it, but in responsive scheduling, it's much more time-consuming.

2.4 Meta-heuristic Algorithms

A meta-heuristic represents a remarkable problem independent algorithmic framework that provides the near optimal solution in polynomial time whereas exact algorithm fails to solve those.

2.4.1 Fruit Fly Optimization Algorithm (FOA)

This paper [29] used the FOA method to predict RNA secondary structure. For forecasting, they have designed four operators based on FOA. Those operators have been designed to perform local and global searches randomly. After that, they have chosen the resultant RNA secondary structures based on the least free energy calculated using the Gibbs free energy formula.

2.4.2 Particle Swarm Optimization Algorithm (PSO)

This algorithm uses a set-based technique for predicting RNA secondary structure [30]. The method's primary purpose is to increase the number of stems for a specific RNA sequence. Then they analyzed of the minimal free energy. The process consists of two stages: identify the level for each swarm at the initial stage, then move ahead to the next stage. They also demonstrated the Kruskal–Wallis test for testing the validity of the hypothesis based on post-hoc analysis. Because of complication of estimating RNA structure they have not done experiment on pseudoknotted RNA sequences.

2.4.3 Chemical Reaction Optimization (CRO)

CRO is a population-based meta-heuristic algorithm based on chemical reaction concepts. The authors in [11] proposed a CRO-based prediction algorithm to predict RNA secondary structure from the primary sequence. First, they generated solution space as a population and choose a probable sequence. Then they randomly decided to perform unimolecular or intermolecular collision operations based on the pre-define MoleColl value. They designed four operators to perform CRO-based RSSP functions.

3 Butterfly Optimization Algorithm

The Butterfly optimization algorithm (BOA) is a natural-inspired algorithm utilizing a population-based approach. The BOA was first introduced by Aroa and Sing [31] in 2019. It is a population based meta-heuristic algorithm that mimics the behaviors of natural butterflies. The capacity of a butterfly is to find food was the primary inspiration for this algorithm. Butterflies possess the highest smelling sense principle, allowing them to locate food from great distances and distinguish between distinct scents within a specific area [29]. The primary approach of the BOA optimization algorithm is foraging, which involves using their sense of smell to find food. In BOA, it is assumed that butterflies produce a smell of certain intensity. Butterflies continue on their way, using the phase as a global search point by detecting the scent of the other. Local search optimization is frequently referred to as a butterfly movement. Random generation is used to accomplish local and global searches. The BOA approach is founded on a trade-off between the smell and scent senses [14]. BOA is a very efficient algorithm with low complexity and a high degree of solving convergence.

3.1 Objective Function for RSSP

The secondary structure of RNA is delineated by a list of base pairs formed from its primary sequence. Let $S = s1, s2, …, sn$ represent an RNA sequence, where S is a string composed of alphabets {a, u, g, c}. A pair (p, q) is termed a base pair (complementary) if {p, q} equals {a, u} or {g, c}. Pairs such as {a, g} and {c, u} are not recognized as base pairs. Among these, the most stable and common base pairs include {g, c}, {a, u}, and {g, u}, along with their counterparts: {c, g}, {u, a}, and {u, g}. Once all these pairs are formed, the RNA strand folds back upon itself, giving rise to its secondary structure.

Our primary objective is to maximize the number of stems in order to construct an RNA secondary structure from a given sequence and select the most stable secondary structure. The secondary structure can be determined

for an individual sequence using thermodynamic principles. These thermodynamic methods predict the stability of a structure and rely on nearest neighbor rules. The stability of a structure can be quantified by calculating the minimum free energy. The structure with the lowest free energy is considered the most stable.

In this method, we have established an objective function, as presented in Eq. (1), based on the individual nearest-neighbor hydrogen bond model (INN-HB), which is a subset of thermodynamic models [11]. The free energy of each helix is calculated using Eq. (2).

$$F = \min\{\Delta G_k\}; 1 \leq k \leq n;$$
$$n = \text{the number of secondary structure in one sequence} \quad (1)$$

$$\Delta G_{37}^{\circ} = \Delta G_{37init}^{\circ} + \sum [\Delta G_{37NN}^{\circ}] + \Delta G_{37AU/GUend}^{\circ}(perAU/GUend) + \Delta G_{37sym}^{\circ} \quad (2)$$

The alternative method for RNA structure prediction that we have used, known as the maximum expected accuracy structure, is determined by the maximum sum of pairing probabilities. Each individual secondary structure prediction sequence is suitably appropriate. The graphical representations of secondary structure prediction problem are shown in Fig. 1 (Table 1).

3.2 Algorithm Design for RSSP

To predict the RSSP issue, we used BOA, a meta-heuristic swarm optimization algorithm. Since the main feature of this algorithm is that it can search both local and global solutions simultaneously. This is a very efficient algorithm for various numerics optimal in problems for its high degree of solving convergence. This method can be efficient and to solve RSSP problem as it needs both global and local search simultaneously. The method is simple so implementation complexity is low. BOA has followed three steps to perform the optimization operation: initialization, iteration, and final stage. The objective function of the RSSP and the solution space are designs at the first initialization phase of the BOA

Table 1 Symbol table for Eq. (2)

Symbol	Meaning
$\Delta G_{37init}^{\circ}$	It is a constant and used when the primary base pair is constructed and initiation occurs
$\sum[\Delta G_{37NN}^{\circ}]$	It is a sum of all base pairs
$\Delta G_{37AU/GUend}^{\circ}$ (perAU/GUend)	It is applied at the end of a helix, per each AU or GU pair
ΔG_{37sym}°	It is a consistency correction. 0.43 kcal/mol for self-completing duplexes and otherwise 0

algorithm. The parameters used in the BOA are assigned at the initialization phase. The algorithm produces to create an initial population of butterflies after assigning initial parameter setting. During the operation of the BOA the total number of butterflies remains same to the initial number for that reason the memory size of the information are fixed [31]. Then new butterflies position are selected randomly from the search space at this time the new butterflies fragrance value and fitness value are calculated and sorted the values. After the initialization phase, the iteration phase creates new artificial butterflies applying different operators of the BOA and also calculated the fitness and fragrance values. After the iteration phase when stopping criteria meet the BOA return best solution. Here's the framework we have design for solving RNA secondary structure prediction problem:

Problem Formulation: Predicting the RNA secondary structure, that includes base pairings, for a given RNA sequence. We have design the objective function using Gibbs's minimum free energy calculation for choosing stable structure with free energy.
Encoding: The primary RNA sequences are represented as strings of nucleotide bases (A, U, G, C), and secondary structures are represented using dot-bracket notation.
Butterfly Optimization Integration: We modify the BOA search and update mechanisms to optimize the RNA secondary structure prediction.

Fig. 1 Folding of three types of structure of RNA

primary (1°) sequence   secondary (2°) structure   tertiary (3°) structure
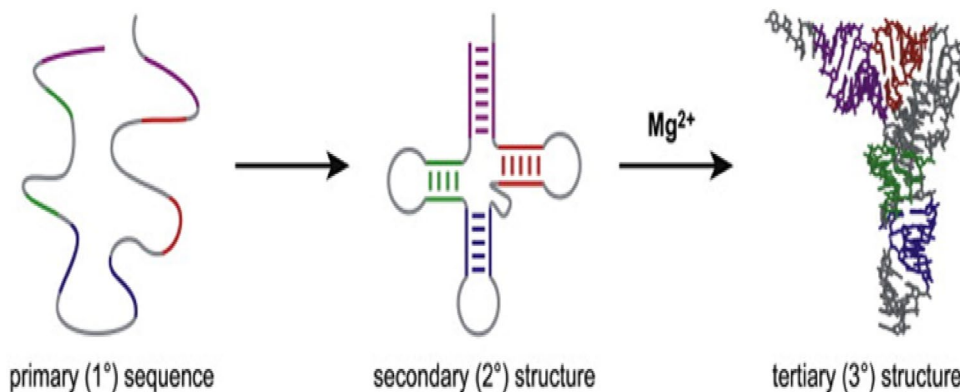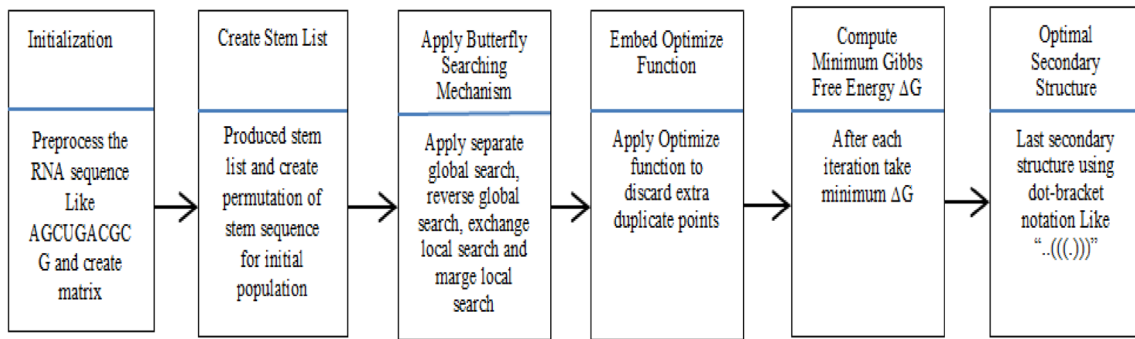
Fig. 2 Block diagram of RNA secondary structure problem

Initialization: The initial solutions (potential RNA secondary structures) are generated by butterfly generation operation creating $n \times n$ matrix of RNA sequences. Butterfly optimization starts with this set of initial solutions.
Search Process: We have developed two global search mechanisms and two local search mechanisms. The separate global search divide each molecules or structure into two structures inject random elements and the reverse global search combine two different region local minima point to explore to search global minimum structure. The exchange local search exchanges different monomer position in a local region of the search space the marge local search chose two different structures in a local region area and marge the even and odd position between this two structures to create local search minima points.

Fig. 3 **A** Sequence matrix from the RNA primary sequence. **B** Stem list from sequence matrix. **C** Permutation of the stem sequence from the Stem list. **D** Randomly selected a stem sequence from Permutation of stem sequence

		00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
		U	G	U	A	G	C	G	U	C	G	U	A	U	G	A	G	C
00	U																	
01	G	1																
02	U	0	1															
03	A	1	0	1														
04	G	1	0	1	0													
05	C	0	1	0	0	1												
06	G	1	0	1	0	0	1											
07	U	0	1	0	1	1	0	1										
08	C	0	1	0	0	1	0	1	0									
09	G	1	0	1	0	0	1	0	1	1								
10	U	0	1	0	1	1	0	1	0	0	1							
11	A	1	0	1	0	0	0	0	1	0	0	1						
12	U	0	1	0	1	1	0	1	0	0	1	0	1					
13	G	1	0	1	0	0	1	0	1	1	0	1	0	1				
14	A	1	0	1	0	0	0	0	1	0	0	1	0	1	0			
15	G	1	0	1	0	0	1	0	1	1	0	1	0	1	0	0		
16	C	0	1	0	0	1	0	1	0	0	1	0	0	0	1	0	1	

(A)

Stem List
Sequence -> (Start, End) -> Length
1 -> (07, 14) -> 4
2 -> (00, 13) -> 4
3 -> (05, 13) -> 3
4 -> (00, 11) -> 3
5 -> (04, 10) -> 3
6 -> (00, 06) -> 3

(B)

Permutation stem sequence
1->6->4->5->2->3
3->2->1->5->6->4
6->2->3->4->5->1
2->4->5->1->3->6
6->5->4->3->1->2

(C)

Randomly selected a stem
6->2->3->4->5->1

(D)

Optimize Function: We developed an optimize function to discard extra duplicate point among the structure and that's speed up the searching procedure. Duplicate point makes the structure to invalid structure and decrease the throughput.

Final stage: In this stage, we choose the best and fittest butterfly from the whole solution and choose the optimum global best stable structure to the next iteration process.

Fine-Tuning and Experimentation: We experiment with different parameters and strategies to enhance the accuracy and efficiency of our integrated method.

3.2.1 Initialization

At first, we have assigned the initial parameters of our proposed algorithm. Then perform the initial population of butterfly generation operation. To generate populations, we take an RNA primary sequence of *n* length. Then we make an $n \times n$ matrix where the row and column for the matrix indicate the corresponding nucleotides of the taken RNA sequence. When an element of the lower triangular matrix indicates the (A-U), (G-C), or (G-U) base pair then the element of the matrix we fill as '1' otherwise '0'. After completion of the filling process, we perform the matrix element '1' searching operation from the bottom left to the top right of the element of lower triangular matrix diagonals. If the number of '1' in a diagonal is more than two, then we consider it as a stem otherwise we skip this. After completing the stem operation, we made a stem list and store the information of the stem such as start position, end position, and length of the stem. By using the total number of stems, we perform the permutation operation and make a permutation list. Then we randomly select a permutation numbers sequence from the permutation list as a butterfly. This whole process is shown in Fig. 2 and the population generation process has shown in Algorithm 1.

Formulate RSSP problem into BOA, first form this matrix in Fig. 3a and from its stem list generate a permutation stem sequence by permutation on the stem list sequence numbers that shown in Fig. 3b, c.

Next, we set the butterfly's parameters. To perform this work, we made a "Butterfly_BOA" class with some attributes to set the parameters such as Ψ, FE, MinStruct, MinFE. Where Ψ is a randomly generated solution from a set of generated solution spaces. FE is Gibbs free energy (ΔG).

Algorithm 1: Population generation

```
Input: RNA primary sequence, populations
1:      for p ← 0 to length(sequence) - 1 do
2:        for q ← 0 to p do
3:          if q<p then
4:            if sequence[p] and sequence[q] are valid base pair then
5:              Matrix[p,q] ← 1
6:            else
7:              Matrix[p,q] ← 0
8:            end
9:          end
10:       end
11:     end
12:     for p ← (length(sequence) - 1) to 1 do
13:       for q ← 0 to (length(sequence) - 3) do
14:         if diagonal '1' found then
15:           Store the stem value such as start, end, length in a list
16:         end
17:       end
18:     end
19:     for p ← 1 to count(populations) do
20:       Sequence ← random value between 1 to count (stem list)
21:       Add this sequence in a butterfly list
22:     end
```

Fig. 4 Separate global search

3.2.2 Iteration
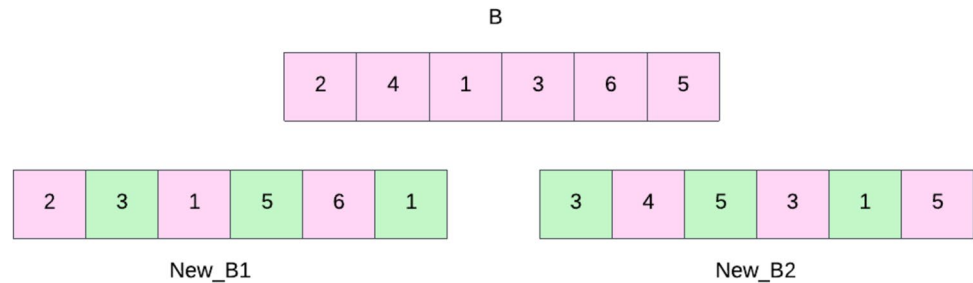
"Butterfly_BOA" is also essential to generate the initial population of butterflies based on assigned population_size. Three methods are involved with this class: the first one is the constructor of this class which constructs and sets the butterfly information. The other two methods are engaged in the search operation. Butterfly_BOA class is presented in Algorithm 2.

Algorithm 2: Class: Butterfly_BOA

```
 1:    Butterfly_BOA()
 2:    {
 3:      randomly generated solution Ψ
 4:      FE ← f(Ψ)
 5:      MinFE ← FE
 6:      MinStruct ← Ψ
 7:    }
 8:    Global_Search()
 9:    Local_Seach()
10:    end
```

In this step, we perform the searching operation to analyze and modify the generated solution space by using the BOA algorithm. To accomplish this operation, we design four operators according to the BOA global and local search concept base. Separate and reverse operators are designed to perform global searches and exchanges, and marge operators are designed to achieve a local search. To select the search operation, we generate a random value between 0 and 1. If the random value is less than the switching probability (P) value, then we perform the global search; otherwise, we perform a local search. To achieve the global search, we select the global fittest butterfly, and to serve the local, we choose the butterfly randomly. Finally, we randomly select the operators based on the search operation.

3.2.2.1 Separate Global Search We choose the recent global best butterfly 'B' as an input from the generated population solution space. Then we separate the butterfly 'B' and create two new butterflies, New_B1 and New_B2, based on the 'B' butterfly's even and odd position values without changing the exact position. Then other places of the new butterflies are filled using random numbers between 1 and 'B'. The procedure is presented in Algorithm 3; an example is shown in Fig. 4.

Algorithm 3: Separate_Global_Search(B)

```
Input: Butterfly B[1,………,n]
Output: New_B1, New_B2
 1:       for p ← 1 to n do
 2:            Generate a random variable x between 1 to n
 3:          if p is odd then
 4:            New_B1[p] = B[p]
 5:            New_B2[p] = x
 6:          else
 7:            New_B1[p] = x
 8:            New_B2[p] = B[p]
 9:          end
10:       end
11:       Optimize (New_B1)
12:       Optimize (New_B2)
```

3.2.2.2 Reverse Global Search To perform this operation, we choose the recent fittest butterfly, B and B', then generate a new butterfly, New_B, whose length is as same as the B. We randomly fill the New_B butterfly value using randomly B butterfly or B' butterfly to reverse the value of the same position. Algorithm 4 presents the reverse global search mechanism, where an example is depicted in Fig. 5.

Algorithm 4: Reverse_Global_Search(B,B')

```
Input: Butterfly B[1,………,n]
Output: New_B
1:      for p ← 1 to n do
2:          Generate a random variable x between 0 to 1
3:          if x < 0.5 then
4:              New_B[p] = B[p]
5:          else
6:              New_B[p] = B'[n-p+1]
7:          end
8:      end
9:      Optimize (New_B)
```

3.2.2.3 Exchange Local Search We choose two random butterflies, B1 and B2. Then these butterflies are divided into two parts, which are used to generate new butterflies. Here, the first part of butterflies B1 and B2 are used to create new butterflies, New_B1, and the last part of butterflies, B1 and B2, are used to create a new butterfly, New_B2. The first part of butterfly B1 and the last part of butterfly B2 are used to generate new butterfly New_B3, and the first part of butterfly B2 and the last part of butterfly B1 are used to

create new butterfly New_B4. The new butterflies' length is the same as butterflies B1 and B2. The process of 'exchange local search' is presented in Algorithm 5, and a corresponding example is shown in Fig. 6.
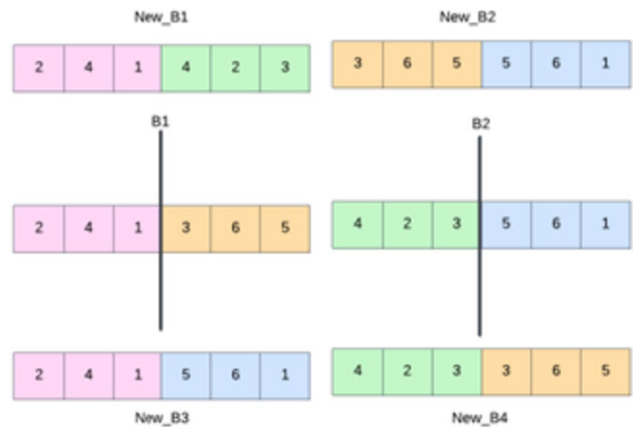

Fig. 6 Exchange local search

Fig. 5 Reverse global search
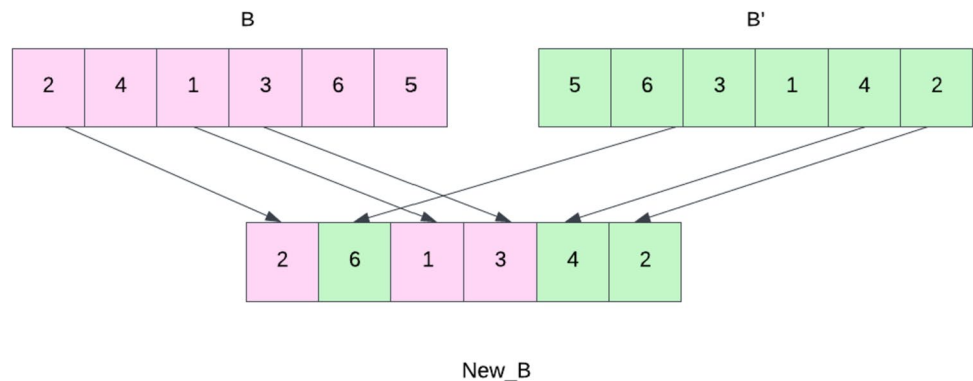
Algorithm 5: Exchange_Local_Search(B1,B2)

```
Input: Butterfly B1[1,………,n],B2[1,………,n]
Output: New_B1, New_B2, New_B3, New_B4
 1:      Set mid ← n/2
 2:      for p ← 1 to mid-1 do
 3:         New_B1[p] = B1[p]
 4:         New_B2[p] = B2[p]
 5:         New_B3[p] = B1[p]
 6:         New_B4[p] = B1[mid+p-1]
 7:      end
 8:      for p ← mid to n do
 9:         New_B1[p] = B2[p]
10:         New_B2[p] = B1[p]
11:         New_B3[p] = B2[p-mid+1]
12:         New_B4[p] = B2[p]
13:      end
14:      Optimize (New_B1)
15:      Optimize (New_B2)
16:      Optimize (New_B3)
17:      Optimize (New_B4)
```

3.2.2.4 Marge Local Search To perform this local search, we randomly select two butterflies, B1 and B2. Then we create two new butterflies, New_B1 and New_B2. For New_B1, we fill up the odd and even position values from the odd position value of B1, and the even position of B2, respectively. Again, for New_B2, we fill up the even and odd position values from the even position value of B1, and the odd position value of B2, respectively. This procedure and its corresponding example are represented in Algorithm 6 and Fig. 7, respectively.

Algorithm 6: Marge_Local_Search (B1, B2)

```
Input: Butterfly B1[1,………,n],B2[1,………,n]
Output: New_B1, New_B2
1:        for p ← 1 to n do
2:           if p is odd then
3:              New_B1[p] = B1[p]
4:              New_B2[p] = B2[p]
5:           else
6:              New_B1[p] = B2[p]
7:              New_B2[p] = B1[p]
8:           end
9:        end
10:       Optimize (New_B1)
11:       Optimize (New_B2)
```

3.2.2.5 Optimize Butterfly Function The butterfly function is an objective function that is used to optimize and decrease the time complexity. It is also essential to speed up the work process. We implement this function to remove extra duplicate stem numbers. To remove identical stem numbers, we have used a 'F[p]' indicator where 'F' is the same length as 'B'. Initially, we set all the index values of F[p] as 0. Then we check the 'F' array values within a loop, if the array values are 0, then we are assigned 1 as the value of the F[p] index, but when the array values are shown 1 that means it's the duplicate stem number, then we have removed it and decrease the array size. The procedure and a corresponding example are shown in Algorithm 7 and Fig. 8, respectively.

Fig. 7 Marge local search
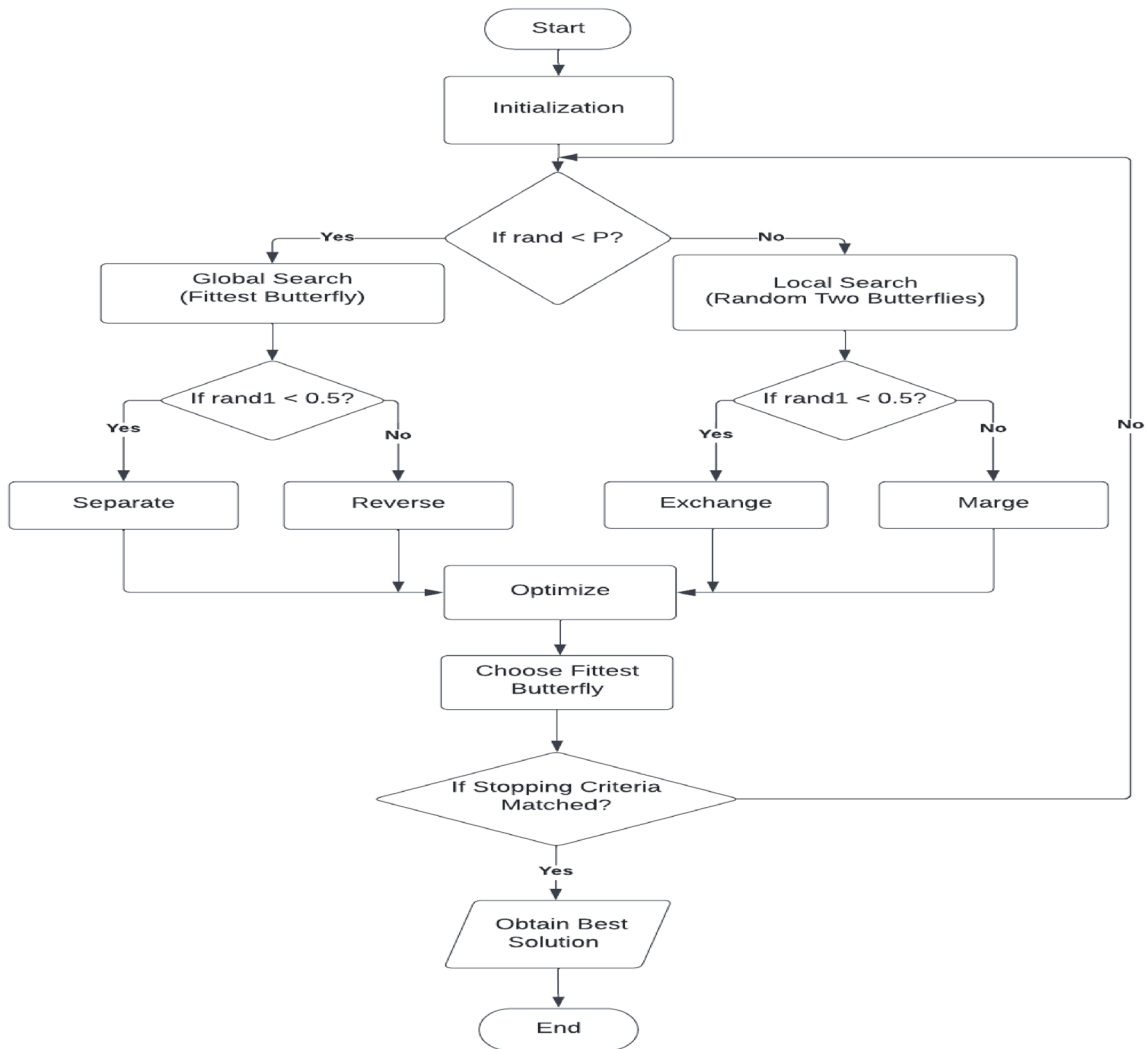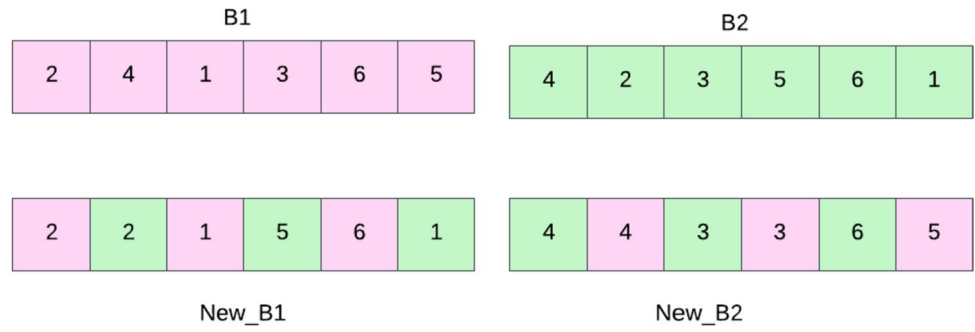
Fig. 8 Optimize function

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Primary	U	G	U	A	G	C	G	U	C	G	U	A	U	G
Initial	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	(	(	(	.	.	.	.	.	.	)	)	)	.	.
3	.	.	.	.	.	.	.	.	(	(	(	)	)	)
2	.	.	.	.	.	(	(	(	.	.	.	)	)	)
3	Remove													
1	(	(	(	(	.	.	.	.	.	.	)	)	)	)
6	(	(	(	.	)	)	)	.	.	.	.	.	.	.
Secondary	**(**	**(**	**(**	**.**	**.**	**.**	**.**	**.**	**.**	**)**	**)**	**)**	**.**	**.**

Fig. 9 Flowchart of BOA

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Primary	U	G	U	A	G	C	G	U	C	G	U	A	U	G
Initial	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	(	(	(	.	.	.	.	.	.	)	)	)	.	.
3	.	.	.	.	.	.	.	.	(	(	(	)	)	)
2	.	.	.	.	.	(	(	(	.	.	.	)	)	)
3	Remove													
1	(	(	(	(	.	.	.	.	.	.	)	)	)	)
6	(	(	(	.	)	)	)	.	.	.	.	.	.	.
Secondary	**(**	**(**	**(**	**.**	**.**	**.**	**.**	**.**	**.**	**)**	**)**	**)**	**.**	**.**

Fig. 10 Construct RNA secondary structure

Algorithm 7: Optimize(B)

```
Input: butterfly B[1,………,n]
Output: Op_B
 1:        for p ← 1 to length(B) do
 2:           F[p] ← 0
 3:        end
 4:        for p ← 1 to length(B) do
 5:           if F[B[p]] = 0 then
 6:              F[p] ← 1
 7:           else
 8:              Remove B[p]
 9:              p ← p-1
10:           end
11:        end
```

Table 2 Parameters of BOA for RNA secondary structure prediction problem

Name	Value
Iteration	80 (max)
Population size (Pop)	150
Switch probability (P)	0.8
Suitability function (F.E)	Objective function

Table 3 Base pair checking matrix

Known structure (PN) versus predict structure (P'N')

	Positive (P)	Negative (N)
Positive (P')	P'P = TP	P'N = FP
Negative (N')	N'P = FN	N'N = TN

3.2.2.6 Final Stage The process is terminated after the completion of the initialization and iteration. In this stage, we choose the best stable solution from the whole solution and give the problem solution as an output. The entire process of the BOA algorithm is shown in Fig. 9.

3.3 Construct RNA Secondary Structure

To build an RNA secondary structure, we need a sequence of stem numbers. Before operating, we ensure that the duplicate stem numbers are removed. Initially, we fill with '.' the n length of the primary sequence corresponding secondary structure. Then we compare every index value of the stem numbers sequence with the stem list to collect the index stem numbers corresponding values, such as start position (S), end position (E), and length (L). Then we use these stem list values to build the index stem number's corresponding secondary structure. To make this structure we fill with '(' according to the start position of the stem number, and the process will continue to increase orderly based on the length of the stem number. Again, according to the end position of the stem number we filled with ')', and this process will continue to decrease based on the length of the stem number. After building all stem index numbers corresponding to secondary structures, we merge all the secondary structures and represent the final secondary structure. This process is shown in Fig. 10.

4 Experimental Results

To solve this problem, we use a public dataset RNA STRAND v2.0 [32]. This dataset contains 20 RNA sequences, and the details of this dataset are presented in "Appendix". To operate, setting the initial parameter for our proposed algorithm is mandatory, as shown in Table 2.

Table 4 Results of BOA, SA[3] and GA[34] in terms of TP, FP, FN and Sensitivity

Sl. no.	TP			FP			FN			Sensitivity		
	SA	GA	BOA	SA	GA	BOA	SA	GA	BOA	SA	GA	BOA
1	22	25	35	20	10	1	16	15	3	57.9	60.5	92.11
2	33	33	34	4	6	0	4	4	3	89.1	89.2	91.89
3	20	10	39	26	29	0	20	30	1	50	25.0	97.5
4	27	27	36	3	3	0	11	11	2	71	71.1	94.74
5	26	33	39	23	3	0	14	7	1	65	82.5	97.5
6	18	25	37	21	8	0	22	15	3	45	62.5	92.5
7	67	75	97	52	46	28	53	45	23	55.8	62.5	80.83
8	64	86	103	126	51	36	62	40	23	50.8	68.3	81.75
9	48	55	96	90	80	40	67	60	19	41.7	47.8	83.48
10	67	68	95	64	63	36	46	45	18	59.2	60.2	84.07
11	74	79	118	88	82	52	64	59	23	53.6	57.2	83.69
12	79	81	114	100	80	48	52	50	17	60.3	61.8	87.02
13	45	63	96	203	90	62	76	58	25	37.2	52.1	79.34
14	43	55	141	160	147	125	146	134	48	22.7	29.1	74.60
15	55	65	180	191	177	119	178	168	53	23.6	27.9	77.25
16	111	74	178	207	154	118	149	186	82	42.6	28.5	68.46
17	103	93	184	133	147	120	148	158	70	41	37.1	72.44
18	111	89	186	146	161	117	155	177	82	41.7	33.5	69.40
19	92	82	188	162	160	116	173	183	77	34.7	30.9	70.94
20	219	224	351	254	412	129	249	235	108	46.7	48.8	76.47

Table 5 Results of BOA, SA[3] and GA[34] in terms of Specificity, F-measure and INF

Sl. no.	Specificity			F-measure			INF		
	SA	GA	BOA	SA	GA	BOA	SA	GA	BOA
1	52.3	69.7	**97.22**	55	64.8	**94.59**	55.03	64.94	**94.63**
2	89.1	84.6	**100**	89.1	86.8	**95.78**	89.1	86.87	**95.86**
3	43.5	25.6	**100**	46.5	25.3	**98.73**	46.64	25.3	**98.74**
4	90	90	**100**	79.38	79.4	**97.29**	79.94	79.99	**97.33**
5	53.0	91.7	**100**	58.4	86.8	**98.73**	58.69	86.98	**98.74**
6	46.2	75.8	**100**	45.6	68.5	**96.11**	45.6	68.83	**96.18**
7	56.3	62	**77.6**	56.05	62.2	**79.18**	56.05	62.25	**79.2**
8	33.7	62.8	**74.1**	40.5	65.4	**77.74**	41.38	65.49	**77.83**
9	34.7	40.7	**70.59**	37.88	44	**76.49**	38.04	44.11	**76.76**
10	51.1	51.9	**72.52**	54.85	55.7	**77.87**	55	55.9	**78.08**
11	65.6	49.1	**69.41**	49.28	52.8	**75.88**	59.3	53	**76.21**
12	44.1	50.3	**70.37**	50.94	55.5	**77.82**	51.57	55.75	**78.25**
13	18.1	41.2	**60.76**	24.4	46	**68.82**	25.95	46,33	**69.43**
14	21.1	27.2	**53**	21.87	28.1	**61.98**	21.89	28.13	**62.89**
15	22.3	26.9	**60.2**	22.93	27.4	**67.67**	22.94	27.4	**68.19**
16	34.9	32.5	**60.14**	38.4	30.3	**64.03**	38.56	30.43	**64.16**
17	43.6	38.8	**60.53**	42.26	37.9	**65.95**	42.28	37.94	**66.22**
18	43.1	35.6	**61.39**	42.39	34.5	**65.15**	42.39	34.53	**65.27**
19	36.2	33.9	**61.84**	35.43	32.3	**66.08**	35.44	32.37	**66.24**
20	46.3	35.2	**73.78**	46.5	40.9	**74.76**	46.5	41.45	**74.78**

Bold values indicate the highest value for each category

Table 6 Results of BOA, COIN method in terms of TP, FP, FN, Sensitivity, Specificity, F-measure and INF

Sl. no.	TP		FP		FN		Sensitivity	
	COIN	BOA	COIN	BOA	COIN	BOA	COIN	BOA
2	33	**34**	1	**0**	4	**3**	89.2	**91.892**
4	33	**36**	0	**0**	5	**2**	86.8	**94.74**
7	82	**97**	35	**28**	38	**23**	68.3	**80.83**
9	59	**96**	66	**40**	56	**19**	51.3	**83.48**
10	71	**95**	46	**36**	42	**18**	62.8	**84.07**
11	81	**118**	68	**52**	60	**23**	57.4	**83.69**
12	80	**114**	83	**48**	51	**17**	61.1	**87.02**
14	50	**141**	91	125	139	**48**	26.5	**74.60**
15	66	**180**	149	**119**	167	**53**	28.3	**77.25**
17	107	**184**	129	**120**	144	**70**	42.6	**72.44**

Bold values indicate the highest value for each category

Table 7 Results of BOA, COIN[35] method in terms of TP, FP, FN, Sensitivity, Specificity, F-measure and INF

Sl. no.	Specificity		F-measure		INF	
	COIN	BOA	COIN	BOA	COIN	BOA
2	97.1	**100**	93	**95.775**	93.07	**95.86**
4	**100**	**100**	93	**97.29**	93.17	**97.33**
7	70.1	**77.6**	69.2	**79.18**	69.19	**79.2**
9	47.2	**70.59**	49.2	**76.49**	49.21	**76.76**
10	60.7	**72.52**	61.7	**77.87**	61.74	**78.08**
11	54.4	**69.41**	55.9	**75.88**	55.88	**76.21**
12	49.1	**70.37**	54.4	**77.82**	54.77	**78.25**
14	35.5	**53**	29.5	**61.98**	30.67	**62.89**
15	30.7	**60.2**	43.9	**67.67**	29.48	**68.19**
17	45.3	**60.53**	56.2	**65.95**	43.93	**66.22**

Bold values indicate the highest value for each category

Table 8 Results of BOA, TL-PSO[30] and FOA[29] method in terms of TP, FP and FN

Sl. no.	TP			FP			FN		
	TL-PSO	FOA	BOA	TL-PSO	FOA	BOA	TL-PSO	FOA	BOA
1	27	32	**35**	7	**1**	**1**	11	6	**3**
2	33	31	**34**	5	1	**0**	4	6	**3**
4	31	35	**36**	5	0	**0**	7	3	**2**
5	36	36	**39**	3	3	**0**	4	4	**1**
14	88	130	**141**	**102**	162	125	94	59	**48**
15	104	161	**180**	127	150	**119**	123	72	**53**
17	122	154	**184**	**113**	133	120	126	100	**70**
18	132	174	**186**	139	146	**117**	128	94	**82**
19	106	164	**188**	127	162	**116**	151	101	**77**
20	276	337	**351**	168	254	**129**	182	122	**108**

Bold values indicate the highest value for each category

Table 9 Results of BOA and TL-PSO[30] method in terms of Sensitivity, Specificity, F-measure and INF

Sl. no.	Sensitivity		Specificity		F-measure		INF	
	TL-PSO	BOA	TL-PSO	BOA	TL-PSO	BOA	TL-PSO	BOA
1	71.1	**92.11**	79.4	**97.22**	75.5	**94.59**	75.14	**94.63**
2	89.2	**91.892**	86.8	**100**	88	**95.775**	87.99	**95.86**
4	81.6	**94.74**	81.6	**100**	83.8	**97.29**	81.6	**97.33**
5	90.0	**97.5**	92.3	**100**	91.1	**98.73**	91.14	**98.74**
14	48.4	**74.60**	46.3	**100**	47.3	**96.104**	47.34	**96.176**
15	45.8	**77.25**	45.1	**53**	45.4	**61.98**	45.41	**62.89**
17	49.2	**72.44**	49.8	**60.2**	49.5	**67.67**	49.5	**68.19**
18	50.8	**69.40**	48.7	**60.53**	49.7	**65.95**	49.74	**66.22**
19	41.3	**70.94**	45.5	**61.39**	43.3	**65.15**	43.35	**65.27**
20	60.3	**76.47**	62.2	**73.78**	61.2	**74.76**	61.24	**74.78**

Table 10 Results of BOA and CRO[11] method in terms of TP, FP, FN, Sensitivity

Sl. no.	TP		FP		FN		Sensitivity	
	CRO	BOA	CRO	BOA	CRO	BOA	CRO	BOA
1	**35**	**35**	2	**1**	**3**	**3**	92.1	**92.11**
2	**34**	**34**	0	**0**	**3**	**3**	91.8	**91.89**
3	38	**39**	1	**0**	2	**1**	95	**97.5**
4	**37**	36	**0**	**0**	1	2	**97.3**	94.74
5	38	**39**	1	**0**	2	**1**	95	**97.5**
6	36	**37**	**0**	**0**	4	**3**	92.3	**92.5**
7	96	**97**	32	**28**	24	**23**	80	**80.83**
8	97	**103**	42	**36**	29	**23**	76.9	**81.74**
9	92	**96**	47	**40**	23	**19**	80	**83.48**
10	91	**95**	39	**36**	22	**18**	80.5	**84.07**
11	113	**118**	59	**52**	28	**23**	80.1	**83.69**
12	107	**114**	61	**48**	24	**17**	81.7	**87.02**
13	91	**96**	70	**62**	30	**25**	75.2	**79.34**
14	129	**141**	156	**125**	60	**48**	68.2	**74.60**
15	170	**180**	144	**119**	63	**53**	72.9	**77.25**
16	166	**178**	143	**118**	94	**82**	63.8	**68.46**
17	169	**184**	156	**120**	85	**70**	66.5	**72.44**
18	174	**186**	147	**117**	94	**82**	64.9	**69.40**
19	176	**188**	149	**116**	89	**77**	66.4	**70.94**
20	345	**351**	144	**129**	114	**108**	75.1	**76.47**

Bold values indicate the highest value for each category

Table 11 Results of BOA and CRO[11] method in terms of Specificity, F-measure and INF

Sl. no.	Specificity		F-measure		INF	
	CRO	BOA	CRO	BOA	CRO	BOA
1	94.5	**97.22**	93.3	**94.59**	93.29	**94.63**
2	**100**	**100**	95.4	**95.775**	95.81	**95.86**
3	97.4	**100**	96.2	**98.73**	96.19	**98.74**
4	**100**	**100**	**98.6**	97.29	**98.64**	97.33
5	**100**	**100**	96.2	**98.73**	97.47	**98.74**
6	**100**	**100**	94.7	**96.104**	94.87	**96.18**
7	75	**77.6**	77.4	**79.18**	77.46	**79.2**
8	69.7	**74.1**	73.2	**77.74**	73.21	**77.83**
9	66.1	**70.59**	72.4	**76.49**	72.72	**76.76**
10	70	**72.52**	74.9	**77.87**	75.07	**78.08**
11	65.6	**69.41**	72.2	**75.88**	72.49	**76.21**
12	63.6	**70.37**	71.5	**77.82**	72.08	**78.25**
13	56.5	**60.76**	64.5	**68.82**	65.18	**69.43**
14	45.2	**53**	54.4	**61.98**	55.52	**62.89**
15	53.3	**60.2**	62.1	**67.67**	62.33	**68.19**
16	53.6	**60.14**	58.1	**64.03**	58.37	**64.16**
17	52	**60.53**	58.3	**65.95**	58.8	**66.22**
18	54.2	**61.39**	59	**65.15**	59.31	**65.27**
19	54.1	**61.84**	59.6	**66.08**	59.94	**66.24**
20	70.5	**73.78**	72.7	**74.76**	72.76	**74.78**

Bold values indicate the highest value for each category

The predicted structure is compared to the known structure to determine the accuracy of the proposed approach. By comparing, we calculate the total number of false-positive (FP), true positive (TP), and false-negative (FN) base pairs. These calculation procedures are shown in Table 3.

To check the effectiveness of the proposed approach, we also calculate the Specificity, Sensitivity, INF, and F-measure as shown in (3), (4), (5), and (6), respectively.

$$Specificity = \frac{TP}{TP + FP} \tag{3}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{4}$$

$$INF = \sqrt{Sensitivity * Specificity} \tag{5}$$

$$F - measure = \frac{2 * Sensitivity * Specificity}{Sensitivity + Specificity} \tag{6}$$

4.1 Experimental Setup

Our algorithm was developed on an Intel(R) Core (TM) i5-8400 processor running at 2.80–2.81 GHz (6 CPUs), with 8 GB RAM and Windows 11 installed (64 bit). We utilized Microsoft Visual Studio Code and C# (6.0.201) for implementation.

4.2 Results Analysis

We apply the proposed approach to solve the RSSP problem and choose the best-predicted structure of fifteen iterations for every RNA sequence. Then we compare the results of our proposed approach (BOA) with the previously developed algorithms. Firstly, a comparison of the BOA with SA[3] and GA[34] is shown in both Tables 4 and 5. In Table 4, we present the results of TP, FP, FN, and sensitivity, whereas specificity, F-measure, and INF are shown in Table 5. The results from Tables 4 and 5 show that BOA outperforms both SA and GA in all the cases.

Then we compare our outcome with the coincidence algorithm (COIN) that was applied in [34]. We use identical sequences to compare BOA and COIN, as shown in Tables 6 and 7. This result indicates that the FP value of COIN is better than BOA only for sequence 14, and BOA is better or at least the same for all other subjects.

We compare our BOA with TL-PSO [30] and FOA [29] for TP, FP, and FN in Table 8, depending on ten specific sequences. The outcomes exhibit that BOA is better than TL-PSO and FOA in all cases except sequences 14 and 17 for FP.

Fig. 11 Graphical comparison results representation of BOA, CRO [11], TL-PSO [18], COIN [34], GA [33], SA [3] in term of **A** Specificity, **B** Sensitivity, **C** INF and **D** F-measure
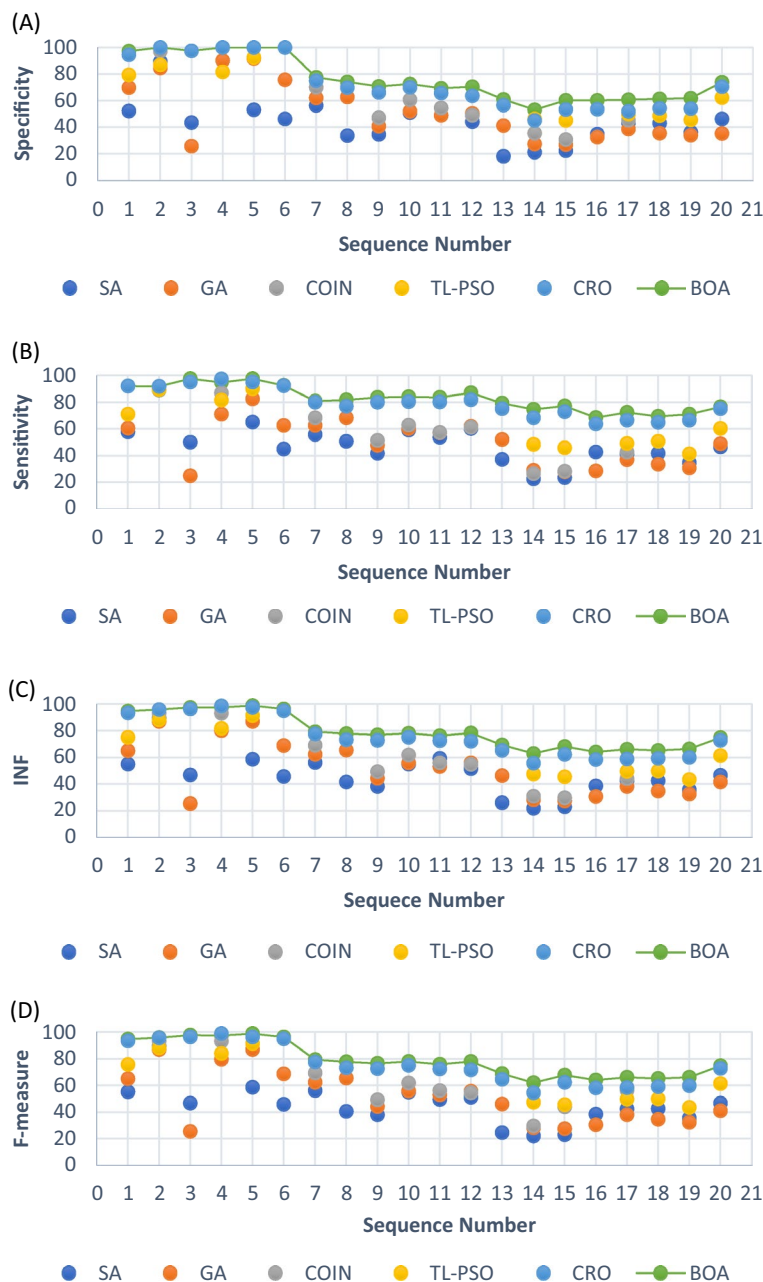
Again, we have shown the test result of TL-PSO[30] and BOA in terms of Sensitivity, Specificity, F-measure and INF in Table 9. In this case, BOA also gives better performance than TL-PSO for all sequence and the best outcome represented in bold text.

Tables 10 and 11 show a comparison of testing twenty RNA sequences between BOA and CRO [11] for different parameters. The results reveal that BOA is better than CRO in most cases, almost for every considered parameter.

Figure 11 shows SA, GA, COIN, TL-PSO and CRO testing results with our proposed BOA based on Specificity, Sensitivity, INF, and F-measure for twenty RNA sequences.

It reveals that the proposed approach is superior to the comparing methods.

The best, worst, average, and standard deviation of each sequence's sensitivity and specificity is shown in Table 12, whereas that of F-measure and INF is shown in Table 13 for the proposed method.

The RNA sequence with the lowest free energy offers the most stable and optimal solution in most cases. We compare our proposed method, BOA, with TL-PSO [30], RNAfold [35], SA [3], and CRO [11] based on the lowest free energy (KCAL/MOL) in Table 14. It reveals that the BOA outperforms all the other methods in all the cases except sequence 2 for SA and sequence 4 for CRO.

Table 12 Simulation results in terms of Sensitivity and Specificity of BOA

Sl. no.	Sensitivity				Specificity			
	Best case	Worst case	Average case	Standard deviation	Best case	Worst case	Average case	Standard deviation
1	92.11	81.57	86.40	3.3	94.59	93.93	96.14	2.2
2	91.89	81.08	86.71	2.93	100	96.77	99.47	1.22
3	97.5	87.5	92.29	3.1	100	94.5	97.79	1.9
4	94.74	89.4	91.67	1.52	100	100	100	0
5	97.5	90	91.88	2.41	100	94.7	97.62	2.79
6	92.5	82.5	87.71	2.71	100	94.3	98.37	2.23
7	80.83	69.1	74.24	3.39	77.6	69.2	72.15	3.1
8	84.92	71.4	77.72	3.5	75.9	64.75	68.66	2.91
9	83.48	69.56	75.34	3.6	70.59	59.26	64.16	3
10	84.07	68.14	77.7	5.3	72.52	59.23	66.71	4.3
11	83.69	72.3	78.47	3.9	69.41	58.62	64.74	3.2
12	87.02	74.04	80.15	3.5	70.37	58.79	64.4	3.4
13	79.34	67.7	72.85	3.3	60.76	51.8	54.68	2.33
14	74.60	63.49	69.57	3.6	53	46.5	52.31	2.7
15	77.25	66.9	73.17	3	63.16	55.7	60.31	2.6
16	68.46	60.8	65.06	2.5	61.38	56.2	60.20	2.1
17	72.44	61.8	67.29	3.4	60.53	56.5	60.47	2.5
18	69.40	60.1	66.45	2.6	65.26	56.9	62.28	2.1
19	70.94	63.7	68.27	2.2	65.73	59.93	62.57	2.1
20	76.47	70.58	74	1.7	73.78	73.13	74.89	1

Table 13 Simulation results in terms of F-measure and INF of BOA

Sl. no.	F-measure				INF			
	Best case	Worst case	Average case	Standard deviation	Best case	Worst case	Average case	Standard deviation
1	93.33	87.32	90.97	2.2	93.34	87.54	91.12	2.1
2	95.77	88.24	92.64	1.98	95.86	88.58	92.87	1.91
3	98.73	90.9	94.95	2.26	98.74	92.2	94.99	2.24
4	97.29	94.4	95.64	0.82	97.33	95.73	95.9	0.79
5	98.73	92.3	94.63	2.15	98.74	92.3	94.63	2.15
6	96.104	87.9	92.72	2.32	96.18	88.2	92.88	2.3
7	79.18	69.2	73.16	3.1	79.2	69.2	73.16	3.1
8	80.15	67.92	72.9	3.1	80.28	68	73.04	3.12
9	76.49	64	69.29	3.25	76.76	64.21	69.52	3.27
10	77.87	63.37	71.75	4.6	78.08	63.53	71.96	4.7
11	75.88	64.76	70.94	3.4	76.21	65.12	71.27	3.5
12	77.82	65.54	71.41	3.4	78.25	65.98	71.84	3.41
13	68.82	58.78	62.47	2.6	69.43	59.31	63.11	2.7
14	61.98	53.69	59.72	3	62.89	54.34	60.33	3
15	69.49	60.8	66.12	2.8	69.85	61.1	66.43	2.8
16	64.73	58.4	62.54	2.3	64.82	58.45	62.58	2.3
17	65.95	59.02	63.69	2.8	66.22	59.1	63.78	2.8
18	67.27	58.44	64.29	2.3	67.30	58.46	64.33	2.3
19	68.24	61.79	65.29	2.1	68.29	61.82	65.36	2.1
20	74.76	71.84	74.41	1.2	74.78	71.89	74.41	1.2

Table 14 Comparison results of BOA, SA [3], TL-PSO [30], RNA-fold [35] and CRO [11] methods based on the lowest free energy (KCAL/MOL)

Sl. no.	TL-PSO	RNAfold	SA	CRO	BOA
1	−47.2	−48.1	−47.04	−57.85	**−58.28**
2	−47.4	−48.3	**−57.52**	−48.89	−51.34
3	–	–	−60.54	−64.14	**−69.47**
4	−53.4	−53.4	−54.94	**−68.8**	−66.59
5	−55.8	−57.1	−58.32	−69.87	**−70.62**
6	–	–	−66.6	−70.56	**−71.16**
7	–	–	−158.54	−173.24	**−173.86**
8	–	–	−178.39	−211.4	**−218.38**
9	–	–	−185.74	−189.68	**−194.66**
10	–	–	−182.06	−183.15	**−193.83**
11	–	–	−223.45	−246.84	**−250.85**
12	–	–	−208.45	−207.32	**−219.32**
13	–	–	−197.23	−217.05	**−228.29**
14	−119.2	−126.8	−175.82	−255.31	**−257.23**
15	−132.0	−136.4	−201.11	−253.59	**−262.84**
16	–	–	−228.3	−344.21	**−352.5**
17	−221.4	−233.1	−305.83	−380.63	**−386.93**
18	−207.4	−222.8	−299.66	−383.63	**−391.92**
19	−176.8	−196	−284.02	−353.97	**−358.95**
20	−702.8	−779.6	−709.08	−784.91	**−794.34**

Bold values indicate the highest value for each category

Table 15 Abbreviation table

RNA	Ribonucleic acid
BOA	Butterfly optimization algorithm
A	Adenine
U	Uracil
G	Guanine
C	Cytosine
NMR	Nuclear magnetic resonance
DP	Dynamic programming
RSSP	RNA secondary structure problem
ACO	Ant colony optimization
GA	Genetic algorithm
CRO	Chemical reaction optimization
SA	Simulated annealing
FOA	fruit Fly optimization algorithm
PSO	Particle swarm optimization algorithm
COIN	Coincidence algorithm
FP	False positive
FN	False negative
TP	True positive

The Butterfly Algorithm has demonstrated its suitability in addressing combinatorial optimization challenges. When it comes to RNA secondary structure prediction, the task essentially revolves around determining the optimal arrangement of stems. This is where the Butterfly Algorithm (BOA) comes into play. The results presented in this section illustrate the strong search capabilities of BOA, outperforming certain meta-heuristic algorithms. However, while BOA can identify optimal solutions for problems, the crux of its effectiveness lies in the fitness function. In the context of the RNA secondary structure prediction problem, the fitness function, representing the energy function, remains imperfect. Consequently, BOA cannot predict all secondary structures with 100% accuracy.

Sequence: UGCCUGGCGGCCGUAGCGCGGUGG UCCCACCUGACCCCAUGCCGAACUCAGAAGUGA AACGCCGUAGCGCCGAUGGUAGUGUGGGGUCUC CCCAUGCGAGAGUAGGGAACUGCCAGGCAU
Benchmark secondary structure:
Dot-parenthesis form: [(((((((((((.....(((((((((....
(((((((.............))))..)))...)))))))).)).((((((..((((((((...))))))
))..)))))))...)))))))))).]
Secondary structure by BOA:
Dot-parenthesis form: [(((((((((((.....(((((((((....
(((((((.............))))..)))...)))))))).)).((((((...((((((()).))))))
))....))))....)))))))))).]

5 Conclusion

In recent decades, the structure of RNA has gotten a lot of interest from researchers because it has a lot of importance in biological issues. This work presents RNA structure optimization and predicts the RNA secondary structure using a swam-based metaheuristic BOA. During the prediction, we calculate the FN, FP, TP, Specificity, Sensitivity, F-measure, and INF that help maintain accuracy. We choose the stable structure based on the structure's least free energy. An additional optimized function has been used that allows for time-consuming and skipping the extra repetition during the solution calculation. The proposed algorithm gives better outcomes and a more stable structure to RSSP problems than other methods. In this work, we have not worked with pseudoknot RNA sequences. In the future, we would like to consider RNA sequences with pseudoknot for solving a similar problem.

Appendix

RNA sequence dataset taken from RNA STRAND v2.0

Sl. no.	Sequence	Accession number	RNA class	Length (nt.)	#Base pair
1	*G. stearother-mophilus*	AJ251080	5 s rRNA	117	38
2	*S. cerevisiae*	X67579	5 s rRNA	118	37
3	*E. coli*	V00336	5 s rRNA	120	40
4	*H. marismortui*	AF034620	5 s rRNA	122	38
5	*T. aquaticus*	X01590	5 s rRNA	123	40
6	*D. radiodurans*	AE002087	5 s rRNA	124	40
7	*M. anisopliae(3)*	AF197120	Group I intron, 23S RNA	394	120
8	*C. saccha-rophila*	AB058310	Group I intron, 23S RNA	454	126
9	*M. anisopliae(2)*	AF197122	Group I intron, 23S RNA	456	115
10	*A. lagunensis*	U40258	Group I intron, 23S RNA	468	113
11	*H. rubra*	L19345	Group I intron, 23S RNA	543	141
12	*A. griffini*	U02540	Group I intron, 23S RNA	556	131
13	*P. leucosticta*	AF342746	Group I intron, 23S RNA	605	121
14	*C. elegans*	X54252	16S RNA	697	189
15	*D. virilis*	X05914	16S RNA	784	233
16	*A. cahirinus*	X84387	16S RNA	940	260
17	*X. laevis*	M27605	16S RNA	945	251
18	*H. sapiens*	J01415	16S RNA	954	266
19	*A. fulgens*	Y08511	16S RNA	964	265
20	*S. acidocal-darius*	D14876	16S RNA	1492	458

All abbreviations are listed in Table 15

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Consent to Participate Not Applicable.

Consent for Publication Not Applicable.

References

1. Arshad R, Fatima I, Sargazi S, Rahdar A, Karamzadeh-Jahromi M, Pandey S, Bilal M. Novel perspectives towards RNA-based nano-theranostic approaches for cancer management. Nanomaterials. 2021;11(12):3330.
2. Pollard TD, Earnshaw WC, Lippincott-Schwartz J, Johnson G. Cell biology E-book. London: Elsevier; 2022.
3. Grypma P, Tsang HH. SARNA-predict: using adaptive annealing schedule and inversion mutation operator for RNA secondary structure prediction. In: IEEE symposium on computational intelligence in multi-criteria decision-making (MCDM); 2014. p. 150–6.
4. Mizuno H, Sundaralingam M. Stacking of crick wobble pair and Watson–Crick pair: stability rules of GU pairs at ends of helical stems in tRNAs and the relation to codon-anticodon Wobble interaction. Nucl Acids Res. 1978;5(11):4451–62.
5. Scarff CA, Thalassinos K, Hilton GR, Scrivens JH. Travelling wave ion mobility mass spectrometry studies of protein structure: biological significance and comparison with X-ray crystallography and nuclear magnetic resonance spectroscopy measurements. Rapid Commun Mass Spectrom Int J Devot Rapid Dissemination Up Min Res Mass Spectrom. 2008;22(20):3297–304.
6. Eddy SR. How do RNA folding algorithms work? Nat Biotechnol. 2004;22(11):1457–8.
7. Kleinkauf R, Mann M, Backofen R. antaRNA: ant colony-based RNA sequence design. Bioinformatics. 2015;31(19):3114–21.

8. Neethling M, Engelbrecht A. Determining RNA secondary structure using set-based particle swarm optimization. In: 2006 IEEE international conference on evolutionary computation. IEEE; 2006. p. 1670–7.

9. Sengupta S, Basak S, Peters R. Particle swarm optimization: a survey of historical and recent developments with hybridization perspectives. Mach Learn Knowl Extract. 2018;1(1):157–91.

10. El Fatmi A, Arakil C, Ali Bekri M, Benhlima S, Sabbane M. A heuristic algorithm for RNA secondary structure based on genetic algorithm. In: 2017 intelligent systems and computer vision (ISCV). IEEE; 2017. p. 1–7.

11. Kabir R, Islam R. Chemical reaction optimization for RNA structure prediction. Appl Intell. 2019;49:352–75.

12. Kai Z, Yulin L. A novel efficient simulated annealing algorithm for the RNA secondary structure predicting with pseudoknots. In: Intelligent computing theories and application: 14th international conference, ICIC, Wuhan, China, Proceedings, Part II 14. Springer International Publishing; 2018. p. 365–70.

13. Tatsuya A. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. Discret Appl Math. 2000;104(1):45–62.

14. Arora S, Anand P. Binary butterfly optimization approaches for feature selection. Expert Syst Appl. 2019;116:147–60.

15. Arora S, Singh S. Node localization in wireless sensor networks using butterfly optimization algorithm. Arab J Sci Eng. 2017;42(8):3325–35.

16. Fan Y, Shao J, Sun G, Shao X. A self-adaption butterfly optimization algorithm for numerical optimization problems. IEEE Access. 2020;8:88026–41.

17. Jalali SMJ, Ahmadian S, Kebria PM, Khosravi A, Lim CP, Nahavandi S. Evolving artificial neural networks using butterfly optimization algorithm for data classification. Neural Inf Process. 2019;596–607:2019.

18. Arora S, Singh S. Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput. 2018;23(3):715–34.

19. Karim MS, Chatterjee S, Hira A, Islam T, Islam R. Protein folding optimization using butterfly optimization algorithm. In: International conference on machine intelligence and emerging technologies. Cham: Springer; 2022. p. 775–87.

20. Zuker M. Mfold web server for nucleic acid folding and hybridization prediction. Nucl Acids Res. 2003;31(13):3406–15.

21. Sato K, Kato Y. Prediction of RNA secondary structure including pseudoknots for long sequences. Brief Bioinform. 2022;23(1):386–95.

22. Sato K, Akiyama M, Sakakibara Y. RNA secondary structure prediction using deep learning with thermodynamic integration. Nat Commun. 2021;12(1):941–50.

23. Chen CC, Chan YM. REDfold: accurate RNA secondary structure prediction using residual encoder-decoder network. BMC Bioinform. 2023;24(1):1–13.

24. Fu L, Cao Y, Wu J, Peng Q, Nie Q, Xie X. UFold: fast and accurate RNA secondary structure prediction with deep learning. Nucl Acids Res. 2022;50(3):e14–e14.

25. Islam MS, Islam MR. A hybrid framework based on genetic algorithm and simulated annealing for RNA structure prediction with pseudoknots. J King Saud Univ Comput Inf Sci. 2022;34(3):912–22.

26. Tong K-K, et al. GAknot: RNA secondary structures prediction with pseudoknots using genetic algorithm. In: 2013 IEEE symposium on computational intelligence in bioinformatics and computational biology (CIBCB). IEEE; 2013. p. 136–42.

27. Liu Y, Hao J, Peng J. Predicting RNA secondary structure with Tabu search. In: 9th IEEE international conference on cognitive informatics (ICCI'10). IEEE; 2010. p. 409–14.

28. Schmitz M, Steger G. Description of RNA folding by "simulated annealing." J Mol Biol. 1996;255(1):254–66.

29. Sajib C, Rabeya SP, Halder S, Mondal M, Sujana FY. RNA secondary structure prediction using fruit fly optimization algorithm. In: 2020 IEEE region 10 symposium (TENSYMP); 2020. p. 1738–42.

30. Lalwani S, Kumar R, Gupta N. An efficient two-level swarm intelligence approach for RNA secondary structure prediction with bi-objective minimum free energy scores. Swarm Evol Comput. 2016;27:68–79.

31. Arora S, Singh S. An effective hybrid butterfly optimization algorithm with artificial bee colony for numerical optimization. Int J Interact Multimed Artif Intell. 2017;4(4):1.

32. Andronescu M, Bereg V, Hoos HH, Condon A. RNA STRAND: the RNA secondary structure and statistical analysis database. BMC Bioinform. 2008;9(1):1–10.

33. Wiese K, Deschenes A, Hendriks A. RnaPredict—an evolutionary algorithm for RNA secondary structure prediction. IEEE/ACM Trans Comput Biol Bioinf. 2008;5(1):25–41.

34. Srikamdee S, Warin W, Prabhas C. RNA secondary structure prediction with coincidence algorithm. In: 2016 16th international symposium on communications and information technologies (ISCIT). IEEE; 2016. p. 686–90.

35. Lorenz R, Bernhart SH, Höner Zu Siederdissen C, Tafer H, Flamm C, Stadler PF, Hofacker IL. ViennaRNA package 2.0. Algorithms Mol Biol. 2011;6(1):1–14.