BIOINFORMATICS ORIGINAL PAPER

Vol. 30 no. 9 2014, pages 1250–1258
doi:10.1093/bioinformatics/btu001

Structural bioinformatics

Advance Access publication January 8, 2014

Evolutionary solution for the RNA design problem

Ali Esmaili-Taheri[1], Mohammad Ganjtabesh[1,2,3,*] and Morteza Mohammad-Noori[1]

[1]Department of Computer Science, School of Mathematics, Statistics, and Computer Science, University of Tehran, P. O. Box: 14155-6455, Tehran, Iran, [2]Laboratoire d'Informatique (LIX), Ecole Polytechnique, 91128 Palaiseau CEDEX, France and [3]School of Biological Science, Institute for Research in Fundamental Sciences (IPM), P.O. Box: 19395-5746 Tehran, Iran

Associate Editor: Ivo Hofacker

ABSTRACT

Motivation: RNAs play fundamental roles in cellular processes. The function of an RNA is highly dependent on its 3D conformation, which is referred to as the RNA tertiary structure. Because the prediction or experimental determination of these structures is difficult, so many works focus on the problems associated with the RNA secondary structure. Here, we consider the RNA inverse folding problem, in which an RNA secondary structure is given as a target structure and the goal is to design an RNA sequence that folds into the target structure. In this article, we introduce a new evolutionary algorithm for the RNA inverse folding problem. Our algorithm, entitled Evolutionary RNA Design, generates a sequence whose minimum free energy structure is the same as the target structure.

Results: We compare our algorithm with INFO-RNA, MODENA, RNAiFold and NUPACK approaches for some biological test sets. The results presented in this article indicate that for longer structures, our algorithm performs better than the other mentioned algorithms in terms of the energy range, accuracy, speedup and nucleotide distribution. Particularly, the generated RNA sequences in our method are much more reliable and similar to the natural RNA sequences.

Availability and implementation: The web server and source code are available at http://mostafa.ut.ac.ir/corna/erd.

Contact: mgtabesh@ut.ac.ir

Received on May 6, 2013; revised on December 29, 2013; accepted on December 31, 2013

1 INTRODUCTION

RNAs perform a wide range of functions in biological systems. The functional form of RNA frequently requires a specific tertiary structure. The scaffold for this structure is provided by secondary structural elements that are hydrogen bonds within the molecule. Therefore, the study and the analysis of RNA secondary structures are crucial to understanding their functional roles (Condon *et al.*, 2004; Higgs, 2000; Hofacker *et al.*, 1994; Khan *et al.*, 2011). In this sense, it is of great interest to propose computational techniques for predicting the RNA secondary structure. Most of the existing computational approaches are based on thermodynamic models that predict the RNA secondary structure with the minimum free energy value (Zuker *et al.*, 1994). Several other approaches are introduced based on maximizing the ensemble probability of the predicted RNA secondary structure (McCaskill, 1990).

An important problem in the RNA research area is the RNA inverse folding problem, in which the secondary structure of an RNA is given and the goal is to find a proper sequence that folds into the given structure. The RNA inverse folding problem can be used to design non-coding RNAs, which are involved in gene regulation, chromosome replication and RNA modification (Cech, 2004; Knight, 2011). The designed sequences are also applicable to construct ribozymes and riboswitches, which may be used as drugs and therapeutic agents in research (Busch and Backofen, 2006; Storz, 2002), or for building self-assembling structures from small RNA molecules in nanobiotechnology (Aguirre-Hernández *et al.*, 2007). In the RNA inverse folding problem, there are exponential number of sequences to be considered as candidates for the solution (Condon *et al.*, 2004; Ganjtabesh and Steyaert, 2011; Haslinger and Stadler, 1999). It is also suggested that the RNA inverse folding problem may be NP-Hard, i.e. finding an exact global solution would require exponential time (Schnall-Levin *et al.*, 2008). Therefore, the heuristic search methods are widely used to address this problem (Aguirre-Hernández *et al.*, 2007; Andronescu *et al.*, 2004; Avihoo *et al.*, 2011; Busch and Backofen, 2006; Dormi *et al.*, 2008; Gao *et al.*, 2010; Hofacker *et al.*, 1994; Ivry *et al.*, 2009; Taneda, 2011; Taneda, 2012).

RNAinverse, available as a part of the Vienna RNA package, is an original approach to solve this problem (Hofacker *et al.*, 1994). This algorithm uses a distance score to measure the distance between the structure of the designed sequence and the target structure. The goal of this algorithm is to minimize the distance score, as well as to maximize the probability of folding the designed sequence into the target structure. When the distance score is 0, the algorithm ends and the generated sequence is returned. The second algorithm, entitled RNA Secondary Structure Designer (RNA-SSD), tries to minimize the structural distance via recursive stochastic local search (Andronescu *et al.*, 2004). Busch and Backofen (2006) proposed another algorithm based on dynamic programming and local search, called INFO-RNA. This algorithm consists of two steps. In the first step, it generates an initial sequence using dynamic programming. In the second step, it uses a stochastic local search method to improve the quality of the initial sequence. Genetic algorithm is also used to solve the RNA inverse folding problem, both for RNA secondary structures (Taneda, 2011) and pseudoknotted structures (Taneda, 2012). Lyngsø *et al.* (2012) introduced a new method

*To whom correspondence should be addressed.

based on a genetic algorithm entitled Frnakenstein, which is implemented in Python. This method allows multiple structures to be specified as target and has a variety of schemes for choosing mutation and recombination operations, or defining fitness and objective functions. The newest version of this method allows pseudoknotted structures to be specified as target or sequence motifs as input, contains parallelization and includes improved searching heuristics as well as multiple folding paradigms. Zadeh *et al*. (2011) used a dynamic programming approach (NUPACK) for designing the RNA sequence that is intended to adopt a target secondary structure at equilibrium. In this method, the sequence design problem is considered as an optimization problem with the goal of reducing the ensemble defect. The ensemble defect (for a sequence and a target secondary structure) is the average number of incorrectly paired nucleotides at equilibrium evaluated over the ensemble of secondary structures. Recently, a constraint programming) approach, entitled RNAiFold, was presented to solve the RNA inverse folding problem. This approach allows a wide range of design constraints to be specified (Garcia-Martin *et al*., 2013). It also introduces a large neighborhood search approach that allows larger instances at the cost of losing completeness, while retaining the advantages of meeting design constraints (motif, GC-content, etc.).

It should be mentioned that the existing methods use a folding algorithm for evaluating and improving the accuracy and the quality of the generated sequences. It is well-known that the *fold* method of the Vienna RNA package requires $O(n^3)$ operations; thus, using this algorithm over the whole sequence increases the overall running time. To resolve this problem, some structural decomposition schemes have been used in the previous studies to reduce the impact of the *fold* method. In this article, a different decomposition scheme is also used to produce the smaller sub-structures, and a new method based on evolutionary algorithms is presented to recursively solve and combine the obtained sub-structures.

Our evolutionary algorithm designs an RNA sequence that folds into a given target structure. Any RNA secondary structure contains different structural components, each having a different length. Therefore, we first reconstruct RNA sub-sequences (pools) corresponding to different components with different lengths. Using the pools, we reconstruct an initial RNA sequence that is compatible (in which the base pairs are canonical) with the given target structure. Then we use an evolutionary algorithm to improve the quality of the sub-sequences corresponding to the components. The major contributions of our algorithm are using the natural RNA sequences, a different method for evaluating the sequences in each population and a different hierarchical decomposition of the target structure into smaller sub-structures.

The rest of this article is organized as follows. In Section 2, the details of our approach are presented. Section 3 represents the results of our approach and the others, as well as the comparisons. Finally, the conclusion is presented in Section 4.

2 APPROACH

In this section, we introduce our evolutionary algorithm for designing an RNA sequence that folds into a given target structure. Any RNA secondary structure can be uniquely decomposed into its structural components (stems, hairpin loops, internal loops, bulge loops and multi-loops), each having a different length (this decomposition is referred to as *component decomposition*). Thus, we first construct the pools of RNA sub-sequences corresponding to different components with different lengths. Using these pools, we then construct an initial RNA sequence that is compatible with the given target structure. After that, the target structure is decomposed into smaller sub-structures (this decomposition is referred to as *hierarchical decomposition*). This decomposition is performed recursively, in positions where the multi-loops occur. Then, for each sub-sequence corresponding to the sub-structure in the hierarchical decomposition, an evolutionary algorithm is used to improve the quality of that sub-sequence. Finally, the improved sub-sequences are combined level by level to yield the result (line 6 of Algorithm 2). The aforementioned steps are summarized in the following algorithms, where T, Is and P represent the target structure, RNA sequences and the population, respectively. The details of these algorithms are presented in the following subsections.

Algorithm 1: Evolutionary RNA Design (T)

1⟩ $I \leftarrow$ Generate-Initial-Sequence (T);
2⟩ return (Hierarchical-Decomposition (T, I));

Algorithm 2: Hierarchical-Decomposition (T, I)

1⟩ **if** ($|multi\text{-}loop(T)| \leq 1$)
2⟩ return (Evolution (T, I));
3⟩ $(T_1, I_1, T_2, I_2) \leftarrow$ Decompose (T, I);
4⟩ $I'_1 \leftarrow$ Hierarchical-Decomposition (T_1, I_1);
5⟩ $I'_2 \leftarrow$ Hierarchical-Decomposition (T_2, I_2);
6⟩ $I \leftarrow$ Combine (I'_1, I'_2);
7⟩ return Evolution (T, I);

Algorithm 3: Evolution (T, I)

1⟩ $best \leftarrow I$;
2⟩ $T_I \leftarrow fold(I)$;
3⟩ **if** ($T \neq T_I$)
4⟩ $P \leftarrow$ Generate-Population (T, T_I, I, $count$);
5⟩ **do**
6⟩ $\{I_1, I_2, I_3\} \leftarrow$ three best solutions from P with respect to free energy;
7⟩ $\{I_1, I_2, I_3\} \leftarrow$ three best solutions among $\{best, I_1, I_2, I_3\}$, increasingly ordered with respect to the Hamming distance;
8⟩ $best \leftarrow I_1$;
9⟩ $P \leftarrow$ Generate-Population (T, T_{I_1}, I_1, $count/3$);
10⟩ $P \leftarrow P \cup$ Generate-Population (T, T_{I_2}, I_2, $count/3$);
11⟩ $P \leftarrow P \cup$ Generate-Population (T, T_{I_3}, I_3, $count/3$);
12⟩ **while** (stopping condition is not satisfied);
13⟩ **end if.**
14⟩ return ($best$);

Algorithm 4: Generate-Population (T, T_I, I, $count$)

1⟩ $Components(T) \leftarrow$ Component-Decomposition (T);
2⟩ $Components(T_I) \leftarrow$ Component-Decomposition (T_I);
3⟩ $D \leftarrow \{c | c \in Components(T) - Components(T_I)\}$;
4⟩ **for** $i \leftarrow 1$ to $count$
5⟩ $c_i \leftarrow$ random element from D;
6⟩ $I_i \leftarrow$ replace the sub-sequence corresponding to c_i in I

with another one, of the same type and length, from the appropriate pool;
7⟩ $E_i \leftarrow energy\_of\_structure(I_i, T)$;
8⟩ $P \leftarrow P \cup \{\langle I_i, E_i\rangle\}$;
9⟩ **end for**
10⟩ return (P);

2.1 Pool reconstruction

To design the RNA sequences that are similar to the natural ones, we use the existing databases of RNA sequences [namely, The RNA secondary STRucture and statistical ANalysis Database (STRAND) (http://www.rnasoft.ca/strand/)] to construct the pools of natural RNA sub-sequences. To do this, for each sequence in the database, the *fold* method of the Vienna RNA package is executed to obtain its secondary structure. Then, using the component decomposition, this structure is decomposed into its structural components and the sub-sequences of the same type and length are gathered into the same pool. The distributions of the number of sub-sequences for different structural components with respect to their lengths are shown in Figure 1, where for longer components, the corresponding pool may contain no sub-sequence. To increase the number of sub-sequences corresponding to the longer components, we artificially generate sub-sequences that fold into a desired structural component. To do this, we first analyze the natural RNA sequences to obtain the distribution of each nucleotide in different components as presented in Table 1. Then the artificial sequences are generated with respect to their types and lengths, as well as the natural distribution of nucleotides. For any longer stem, the sub-sequences of shorter stems are combined to produce a sub-sequence that is folding to that stem. For each longer loop component, an evolutionary algorithm is used to generate the corresponding sub-sequence. All generated sub-sequences are verified by using the *fold* method. For each type and length of the structural component, 300 sub-sequences are generated and stored in the corresponding pool for future use.

2.2 Construct the initial sequence

To be consistent with the other parts of our algorithm, especially when we improve the quality of the sub-sequences corresponding to the components, we assign a compatible RNA sequence to the given target structure (line 1 of Algorithm 1). To do this, the target structure is decomposed into its structural components and based on the type and length of each component, a sub-sequence is randomly picked from the corresponding pool. These sub-sequences are then assembled to produce a compatible sequence for the target structure. It should be noted that this initial sequence is not guaranteed to fold into the target structure and therefore it should be considered for further improvements.

2.3 Hierarchical structure decomposition

Because the *fold* method requires $O(n^3)$ operations, improving the whole initial sequence will increase the overall running time of any heuristic algorithm. On the other hand, decomposing the target structure into its structural components produces many small components and increases the number of iterations, and consequently increases the running time. To speed up the RNA design problem, we use the hierarchical decomposition scheme to decompose the target structure into its sub-structures (line 3 of Algorithm 2). This decomposition is done in positions where the multi-loops occur. Let the given structure consist of k multi-loops $M_1, M_2, \ldots, M_k$. For each p ($1 \le p \le k$), let M_p contain q closing base pairs, say $i_p^1.j_p^1, i_p^2.j_p^2, \cdots, i_p^q.j_p^q$. We define an order over the closing bases as follows:

$$i_p^a.j_p^a < i_p^b.j_p^b \Leftrightarrow i_p^a < i_p^b$$

The minimum closing base pair in each multi-loop is called *tag base pair*, and the stem containing a tag base pair is called *tag stem*. Now, the minimum base pair (in tag stem with respect to the previously defined order) is marked as a *breaking base pair*, where the hierarchical decomposition is performed. The tag base pairs and the breaking base pairs are shown in Figure 2. If several breaking base pairs are available, then the one whose resulting sub-structures have almost equal lengths is chosen. The process of decomposing the given structure is performed recursively to yield the hierarchy over the sub-structures (lines 4–5 of Algorithm 2). The decomposition is continued until a rooted binary tree of k leaves (i.e. $2k-1$ nodes) is obtained. Here, the root node corresponds to the target

Table 1. Natural distribution of each nucleotide in different structural components

	ML	Stem	HP	IL	BL	Total
A	0.4614	0.1626	0.3914	0.4013	0.4371	0.2614
G	0.1942	0.3374	0.2204	0.2682	0.1495	0.2923
C	0.1493	0.274	0.1593	0.1392	0.1533	0.2273
U	0.1952	0.226	0.229	0.1912	0.26	0.2191

Note: Stems are denoted by S, hairpins by HP, multi-loops by ML, internal loops by IL and bulges by BL.

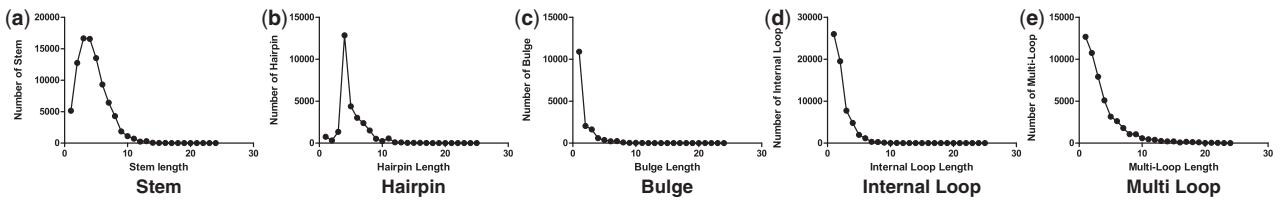

Fig. 1. The distribution of the number of sub-sequences for different structural components with respect to their length

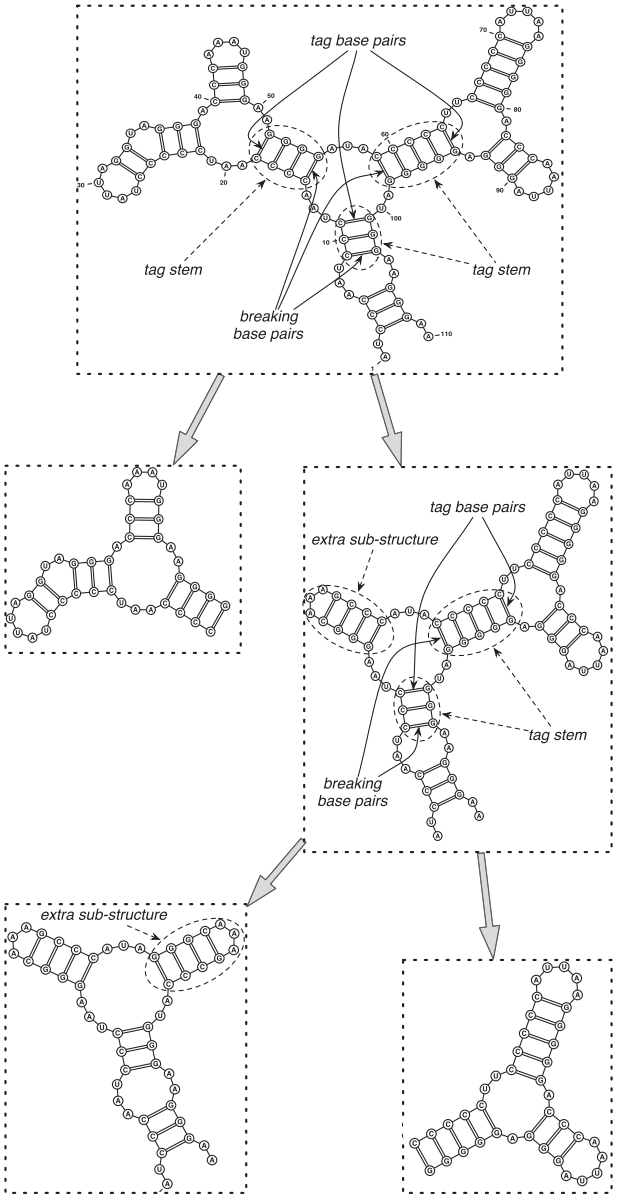

Fig. 2. The hierarchical decomposition of target structure into sub-structures

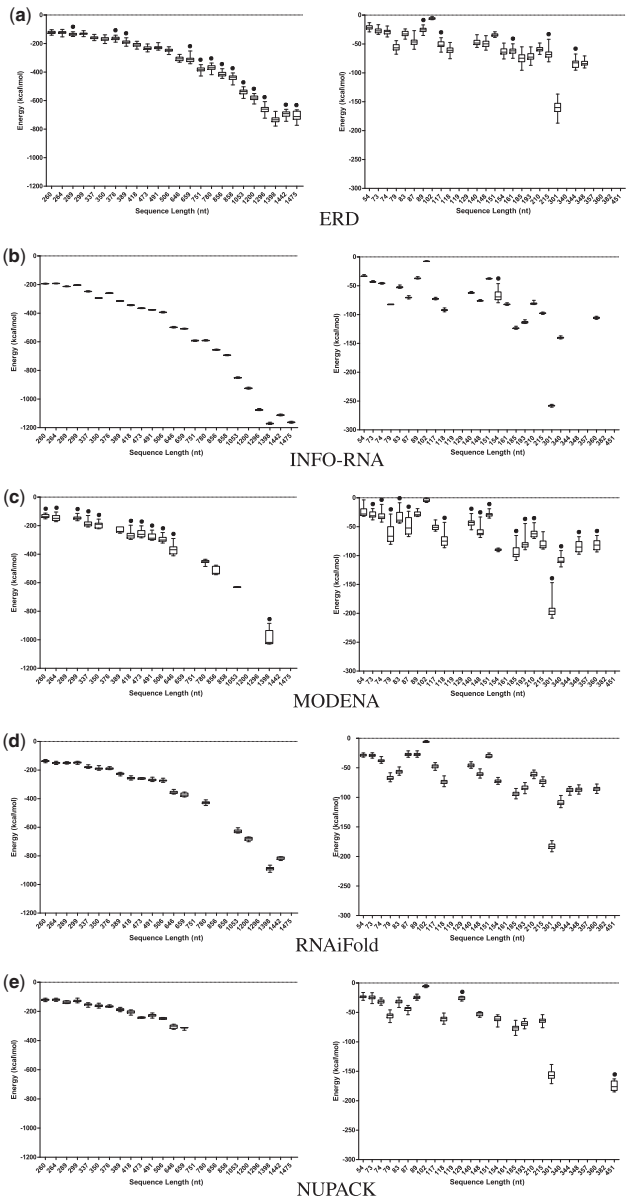

Fig. 3. Energy ranges of ERD, INFO-RNA, MODENA, RNAiFold and NUPACK for dataset A (left) and dataset B (right). The wider energy ranges are specified by the filled circles above them

structure, the leaves correspond to the sub-structures of at most one multi-loop and the intermediate nodes correspond to the sub-structures of more than one multi-loop. To preserve the affinity of the sub-structures at breaking position and to reduce the size of the sub-structures in the hierarchy, the sub-structure starting from any breaking base pair is replaced with a small structure of the form '(((((...)))))' with the fixed sequence 'GGGCAAAGCCC'. This extra sequence is eliminated after generating sub-sequences corresponding to the sub-structures (line 6 of Algorithm 2). The initial compatible sequence is also decomposed exactly at the same positions with respect to the corresponding sub-structure. Our hierarchical decomposition is different from the one proposed in RNA-SSD. While the result of our decomposition is a set of sub-structures, each containing at most one multi-loop, the RNA-SSD decomposition yields a set of sub-structures having lengths between 30 and 70 nt. The sub-structures and corresponding sub-sequences are given as input to our evolutionary algorithm for future improvements.

2.4 Evolutionary algorithm

After constructing the pools of sub-sequences as well as the initial sequence, we use an evolutionary algorithm for each sub-sequence corresponding to the node in the hierarchical decomposition tree to improve its quality (Algorithm 3). The first step in our evolutionary algorithm is the construction of the initial population (Algorithm 4). To do this, we use the *fold* method over the initial sequence to determine its minimum free energy secondary

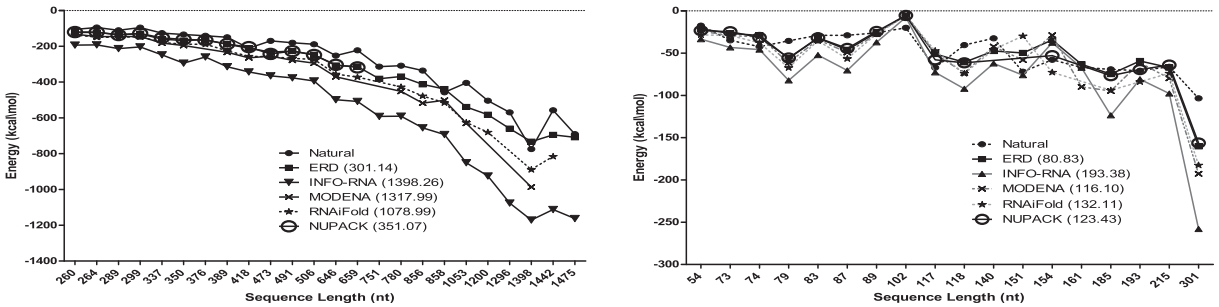

Fig. 4. Average energies of sequences generated by ERD, INFO-RNA, MODENA, RNAiFold and NUPACK with respect to the natural energies for dataset A (left) and dataset B (right). The Euclidean distance is used to measure the closeness of generated sequences to the natural ones as indicated in front of each approach

Table 2. The average Boltzmann probabilities of the structures corresponding to the generated sequences of different methods for dataset A

ID	Length (nt)	Natural	ERD	INFO-RNA	MODENA	RNAiFold	NUPACK
Z83250	260	0.000383888	**0.003606361**	0.018580258	0.054937316	0.179119811	0.487296
L11935	264	5.07E-06	**0.00301916**	0.005283445	0.136243128	0.513156544	0.519628
LIU92530	289	8.37E-05	**0.007972919**	0.008513586	–	0.207993284	0.023660156
U84629	299	1.11E-07	**0.003461317**	0.010093321	0.131891533	0.307077719	0.4592375
AF107506	337	1.77E-06	**0.001504984**	0.006577567	0.049137452	0.251902467	0.421332432
AF106618	350	3.48E-07	**0.00188364**	0.00404454	0.034394623	0.155467862	0.44861
AJ011149	376	1.60E-07	**0.00013367**	0.0012954	–	0.067272428	0.064881688
S70838	389	2.60E-09	0.000181682	**0.000137784**	0.040982974	0.101641998	0.31202676
U63350	418	3.86E-07	**0.000449907**	0.002373525	0.081040715	0.324052388	0.357868182
AF141485	473	1.86E-07	**0.000300153**	0.001986762	0.009796701	0.0766203	0.230875
U81771	491	2.97E-10	**6.16578E-05**	0.0010017	0.021417009	0.184506802	0.276122642
AJ130779	506	4.10E-08	**0.000119147**	0.002082234	0.017317182	0.068433166	0.19845
AF096836	646	3.38E-11	**4.49075E-06**	9.0156E-05	0.00684705	0.10196185	0.212957143
X61771	659	1.97E-12	4.97256E-07	**3.46852E-07**	–	0.008166972	0.053119
AJ236455	751	5.50E-16	4.53431E-06	**3.8221E-07**	–	–	–
AJ132572	780	6.12E-14	4.55262E-08	**2.68058E-08**	0.002966551	0.040116833	–
AB015827	856	1.22E-16	**1.6684E-08**	1.23071E-06	0.000951225	0.00755774	–
D38777	858	5.12E-14	8.13192E-08	**4.54197E-09**	0.00014058	0.0165205	–
AF029195	1053	1.45E-21	**2.45553E-09**	3.44484E-08	6.79024E-06	0.002990993	–
X81949	1200	1.48E-20	**6.4575E-11**	1.0131E-08	–	0.002224721	–
AJ133622	1296	2.02E-20	**9.35269E-12**	7.79333E-11	–	–	–
AF056938	1398	4.44E-23	**4.69271E-13**	7.23494E-11	9.08662E-06	0.002353822	–
X99676	1442	6.17E-29	**2.24016E-15**	1.36979E-11	–	0.000620377	–
L77117	1475	6.24E-28	**5.50428E-14**	4.80856E-12	–	–	–

Note: The closest probabilities to the natural ones are indicated in bold.

structure (line 2 of Algorithm 3). The predicted structure may differ from the target structure in some positions. Then we use the component decomposition scheme to find the components containing these positions of differences (lines 1–3 of Algorithm 4). Among these components, 15 components are selected randomly. For each selected component, its corresponding sub-sequence in the initial sequence is replaced with another subsequence from the appropriate pool, regarding its type and length (lines 5–6 of Algorithm 4). Then the newly obtained sequence is evaluated by using the *energy_of_structure* method (available as a part of the Vienna RNA package) to determine its thermodynamic free energy over the target structure (line 7 of Algorithm 4). Therefore, starting from the initial sequence, 15 sequences are generated and evaluated (*count* = 15 in Algorithm 3). These new sequences are considered as the initial population (line 4 of Algorithm 3). To evaluate the quality of the sequences in the current population, several steps are taken in our algorithm. These sequences are sorted increasingly according to their energy values. Among them, three best sequences (with lowest energies) are selected for further evaluations (line 6 of Algorithm 3). The structures of these three sequences are determined by using the *fold* method. Then these three selected structures, as well as the best structure found, are considered, and the Hamming distances between the target structure and each of these structures are evaluated (line 7 of Algorithm 3). Again, the best three of them (lines 9–11 of Algorithm 3) are chosen as a basis for generating the next population (five sequences are produced from each one). The best one is also stored as the best solution till now (line 8 of Algorithm

3). It should be noted that in our evolutionary algorithm, we do not have crossover operation and the mutation operates at component level (not at nucleotide level). The stopping condition in our algorithm (line 12 of Algorithm 3) is either finding a solution whose Hamming distance with the target structure is 0 or continuing the aforementioned processes for at most 250 iterations (in this case, the final best solution is reported). The most expensive part of our algorithm is the *fold* method that requires $O(n^3)$ operations. This method is performed three times, during each iteration of the Algorithm 3. For a structure of k multi-loops ($k \ll n$ in natural structures), the Evolution function (Algorithm 3) is recursively executed for at most $2k-1$ times because of the decomposition scheme, which yields a rooted binary tree of $2k-1$ nodes. Therefore, the overall time complexity of our algorithm is $3 \times 250 \times (2k-1) \times O(n^3)$. In the worst case, if we consider $k \in O(n)$, the time complexity of our algorithm becomes $O(n^4)$.

Our proposed evolutionary algorithm is different from MODENA and Frnakenstein approaches in three parts. First, we use the pools of natural RNA sequences, whereas MODENA and Frnakenstein use random sequences. Second, we evaluate the sequences by their energies [which requires $O(n)$ operations] and then evaluate three best sequences among them by the *fold* method [which requires $O(n^3)$ operations], whereas MODENA and Frnakenstein use the *fold* method over the whole population to evaluate the sequences that will increase the overall running time. Third, we use the component-based mutation, whereas MODENA and Frnakenstein use the mutation over the nucleotides.

3 RESULTS

To test the accuracy and reliability of our algorithm, we use two different datasets. The first one is the dataset that is used in Andronescu *et al.* (2004) (dataset *A*). This dataset contains 24 sequences with length between 260 and 1475 nt. The structures of these sequences are determined by *RNAfold* and they are used as target structures in our comparison. The second dataset is chosen from Taneda, 2011, which contains 29 structures with length between 54 and 451 nt (dataset *B*). The existence of large loops in the structures of this dataset makes it difficult to design RNA sequences. We compare our algorithm with four other approaches, namely, INFO-RNA, MODENA, RNAiFold and NUPACK. Different measures are used in our comparison, as presented in the following subsections. All the results are obtained by a computer with Core2Duo (2.26 *GHz*) CPU, having 2 GB of memory and running Linux Ubuntu (11.04) as the operating system. The Vienna RNA Package (version 1.8.5) along with the Turner free energy parameters (Mathews *et al.*, 1999) is used in all approaches. Among these approaches, NUPACK is the only one that uses the ensemble defect as a

Table 3. The average Boltzmann probabilities of the structures corresponding to the generated sequences of different methods for dataset B

ID	Length (nt)	Natural	ERD	INFO-RNA	MODENA	RNAiFold	NUPACK
RF00008	54	0.229978	**0.233941634**	0.446077496	0.467788377	0.68505846	0.784268
RF00029	73	0.327886	0.189181972	0.085379142	**0.225253537**	0.524949591	0.66849894
RF00005	74	0.131162	**0.268575738**	0.434650012	0.364912413	0.884831933	0.779874
RF00027	79	0.0347672	0.108456679	**0.060126016**	0.336743728	0.596693714	0.768822
RF00019	83	0.119012	**0.121988045**	0.105659891	0.390602457	0.61254028	0.84314
RF00014	87	0.0557285	**0.23024102**	0.288756114	0.354331389	0.74768358	0.806242
RF00006	89	0.0106282	**0.01087495**	0.005274549	0.339961416	0.717349917	0.82521
RF00026	102	0.0019262	0.206386699	**0.026038938**	0.428731075	0.930153816	0.80229
RF00001	117	0.00279109	0.082466413	**0.078671521**	0.110532411	0.244872	0.066514167
RF00021	118	0.00721032	0.058592087	**0.048354237**	0.283041059	0.54165448	0.645164
RF00020	119	4.51E-03	–	–	–	–	–
RF00016	129	1.35E-02	–	–	–	–	**0.5876465**
RF00015	140	0.0026174	0.006119246	**0.000447298**	0.073475666	0.239242982	–
RF00022	148	0.0020379	0.007856103	**0.003372637**	0.158246813	0.384468773	0.559157143
RF00002	151	0.00153722	0.012590843	**0.005728835**	0.147539767	0.322004353	–
RF00007	154	0.0022317	0.021953073	**0.00619692**	0.145771653	0.335369156	0.532159606
RF00003	161	0.000978519	0.034592406	**0.01966985**	–	–	–
RF00013	185	0.000385769	0.006482949	**0.001664969**	0.206212295	0.84395658	0.664858
RF00004	193	0.00049132	**0.00172956**	0.00366305	0.061473176	0.48222666	0.505296552
RF00025	210	3.69E-05	0.000930857	**0.000134918**	0.208671144	–	–
RF00012	215	0.000303951	0.005821207	**0.000548325**	0.22742912	0.53416718	0.522234783
RF00017	301	4.10E-09	**0.020305168**	0.021128916	0.136416679	0.323599534	0.48431
RF00030	340	1.66E-06	–	**2.95481E-05**	0.021685914	0.343198533	–
RF00028	344	2.30E-06	**3.95058E-06**	–	–	0.198090668	–
RF00009	348	2.85E-06	**1.97457E-06**	–	0.026462644	0.288621144	–
RF00010	357	8.68766E-11	–	–	–	–	–
RF00018	360	9.84E-08	–	**7.5304E-05**	0.018923108	0.330605522	–
RF00011	382	1.62E-08	–	–	–	–	–
RF00024	451	8.15E-07	–	–	–	–	**0.09860725**

Note: The closest probabilities to the natural ones are indicated in bold.

fitness function, where the threshold value is considered 0.01. For generating the results, the constraint programming version of RNAiFold is used without any constraint (default mode). Also in MODENA, the default parameters (50 for both population size and iterations) are used.

3.1 Stability and energy ranges

For any RNA structure, there are many compatible sequences with different levels of free energy. The difference between the highest and lowest free energy structure is considered as the *energy range*. Solutions of wider energy ranges are distributed more diversely in the solution space. From this point of view, our algorithm and MODENA produce sequences with wider energy ranges (using more diverse search) than the other approaches, as is understood from the energy ranges presented in Figure 3 for both datasets (the wider energy ranges are specified by the filled circles above them). This gives us an opportunity to select an RNA sequence in a wider range of free energies. On the other hand, the average energies of the sequences generated by our method are closer to the natural energies, compared with the other approaches, as presented in Figure 4, where the Euclidean distance is used as a measure of closeness. This helps us to select an RNA sequence whose secondary structure has free energy closer to that of natural counterparts. The average Boltzmann probability of the designed sequences is also calculated and presented in Tables 2 and 3 for datasets *A* and *B*, respectively. The closest probabilities to the natural ones are indicated in bold. Dash in a cell indicates that the corresponding method could not generate any sequence for that instance. The Boltzmann probability of a structure S is defined by:

$$p(S) = \frac{exp(-E(S)/RT)}{Z}$$

where Z is the partition function, $E(S)$ is the energy of the generated sequence over the structure S, R is the gas constant and T is the absolute temperature (McCaskill, 1990). Comparing with the other methods, NUPACK is superior in generating sequences that have higher Boltzmann probabilities. However, it should be noted that the Boltzmann probabilities of natural sequences are low, as reported for Evolutionary RNA Design (ERD) and the other methods. The Boltzmann probabilities for ERD are also closer to the probabilities of natural sequences.

3.2 Accuracy and speedup

The accuracy and speed comparisons of our algorithm with other methods are presented in Tables 4 and 5 for datasets *A* and *B*, respectively, where the best results are indicated in bold. For each structure, all mentioned approaches are executed 50 times and the time limit for all approaches is considered 600 s (as it is used in RNAiFold). The success count (*SC*) indicates how often each approach successfully designs an RNA sequence (among 50 executions). The expected time (E_t) indicates how much time is

Table 4. Accuracy and speed comparison between ERD and other methods for dataset *A*

ID	Length(nt)	ERD		INFO-RNA		MODENA		RNAiFold		NUPACK	
		$E_t(s)$	*SC*	$E_t(s)$	*SC*	$E_t(s)$	*SC*	$E_t(s)$	*SC*	$E_t(s)$	*SC*
Z83250	260	1.52	50	**1.083**	**50**	207.993	31	49.412	47	188.461	50
L11935	264	1.80	50	**1.105**	**50**	236.099	50	3.969	50	62.169	50
LIU92530	289	2.312	50	**1.815**	**50**	∞	0	105.176	44	3879.494	4
U84629	299	**2.376**	**50**	3.201	50	278.059	25	35.421	48	295.932	47
AF107506	337	3.256	50	**2.825**	**50**	285.05	47	334.603	33	323.18	37
AF106618	350	4.098	50	**0.977**	**50**	337.721	46	130.228	42	103.792	50
AJ011149	376	3.308	50	**1.669**	**50**	∞	0	806.793	22	1248.668	11
S70838	389	6.525	50	**4.509**	**50**	473.889	7	76.935	46	427.562	37
U63350	418	**3.455**	**50**	5.043	50	642.258	45	12.165	50	258.221	44
AF141485	473	5.653	50	**4.540**	**50**	841.379	38	1350.857	16	9400.725	3
U81771	491	4.822	50	**3.639**	**50**	831.858	48	134.014	42	223.725	50
AJ130779	506	5.754	50	**3.536**	**50**	876.77	48	766.628	24	555.602	40
AF096836	646	10.186	50	**9.374**	**50**	1386.58	42	370.261	32	666.858	35
X61771	659	46.952	50	**22.762**	**50**	∞	0	5524.132	5	797.217	30
AJ236455	751	**44.642**	**50**	1384.1	36	∞	0	∞	0	∞	0
AJ132572	780	**16.801**	**50**	62.468	50	2194.032	49	594.421	27	∞	0
AB015827	856	**17.392**	**50**	19.484	50	3386.394	43	29887.149	1	∞	0
D38777	858	**28.378**	**50**	64.594	50	3290.228	1	30052.557	1	∞	0
AF029195	1053	**31.074**	**50**	93.038	50	5489.113	4	2384.559	11	∞	0
X81949	1200	**39.929**	**50**	126.966	50	∞	0	1459.136	17	∞	0
AJ133622	1296	**50.816**	**50**	249.429	50	∞	0	∞	0	∞	0
AF056938	1398	89.360	50	**49.159**	**50**	11957.11	49	1039.827	26	∞	0
X99676	1442	**83.075**	**50**	197.988	50	∞	0	4799.235	6	∞	0
L77117 (157984-159459)	1475	**48.687**	**50**	84.729	50	∞	0	∞	0	∞	0
SUM		**552.181**	**1200**	2398.042	1186	33111.479	573	79917.488	590	18430.889	488

Note: The best results are indicated in bold. The ∞ symbol indicates no result.

Table 5. Accuracy and speed comparison between ERD and other methods for dataset B

ID	Length(nt)	ERD		INFO-RNA		MODENA		RNAiFold		NUPACK	
		$E_t(s)$	*SC*	$E_t(s)$	*SC*	$E_t(s)$	*SC*	$E_t(s)$	*SC*	$E_t(s)$	*SC*
RF00008 Hammerhead_3	54	0.016	50	**0.011**	**50**	41.905	49	0.73	50	7.005	50
RF00029 Intron_gpII	73	0.373	50	**0.033**	**50**	71.502	38	780.849	22	41.415	41
RF00005 tRNA	74	0.045	50	**0.032**	**50**	36.046	47	68.708	45	24.85	50
RF00027 let-7	79	**0.1**	**50**	0.119	50	61.825	40	13.304	49	20.732	50
RF00019 Y_RNA	83	0.136	50	**0.091**	**50**	50.813	35	3.616	50	9.211	50
RF00014 DsrA	87	0.103	50	**0.034**	**50**	63.707	35	0.848	50	6.389	50
RF00006 Vault	89	**0.23**	**50**	0.252	50	37.926	41	117.487	42	51.291	50
RF00026 U6	102	**0.038**	**50**	11.82	33	57.124	44	129.485	49	209.624	50
RF00001 5S_rRNA	117	2.389	49	**0.255**	**50**	53.41	46	914.629	20	25423.278	1
RF00021 Spot_42	118	0.3	50	**0.164**	**50**	59.139	46	0.964	50	18.411	50
RF00020 U5	119	∞	0	∞	0	∞	0	∞	0	∞	0
RF00016 SNORD14	129	∞	0	∞	0	∞	0	∞	0	**4028.944**	**7**
RF00015 U4	140	**2.218**	49	2.885	**50**	77.096	42	316.477	33	∞	0
RF00022 GcvB	148	**2.846**	**50**	6.382	50	71.024	44	89.310	44	1870.129	14
RF00002 5_8S_rRNA	151	**7.831**	49	11.294	**50**	64.602	39	1198.218	17	∞	0
RF00007 U12	154	1.429	50	**0.975**	**50**	85.534	43	177.556	39	531.148	32
RF00003 U1	161	**23.325**	**43**	379.851	14	∞	0	∞	0	∞	0
RF00013 6S	185	3.014	50	8.601	50	104.908	46	**1.593**	**50**	18.1	50
RF00004 U2	193	**3.177**	**50**	5.883	50	84.767	41	40.564	47	814.047	29
RF00025 Telomerase-cil	210	7.584	50	7.361	50	106.894	48	**1.8505**	**50**	∞	0
RF00012 U3	215	**3.164**	**50**	44.89	49	111.521	48	26.3	50	345.226	46
RF00017 SRP_euk_arch	301	4.331	50	**1.373**	**50**	350.1	48	408.861	32	69.3605	50
RF00030 RNase_MRP	340	∞	0	**207.593**	**50**	294.552	36	412.297	30	∞	0
RF00028 Intron_gpI	344	**147.091**	**44**	∞	0	∞	0	780.401	22	∞	0
RF00009 RNaseP_nuc	348	**268.613**	28	∞	0	288.867	**45**	350.707	34	∞	0
RF00010 RNaseP_bact_a	357	∞	0	∞	0	∞	0	∞	0	∞	0
RF00018 CsrB	360	∞	0	296.386	**50**	**293.777**	39	773.648	23	∞	0
RF00011 RNaseP_bact_b	382	∞	0	∞	0	∞	0	∞	0	∞	0
RF00024 Telomerase-vert	451	∞	0	∞	0	∞	0	∞	0	**1387.425**	**12**
SUM		**478.362**	**1062**	986.298	1046	2467.039	940	6608.411	898	34876.594	682

Note: The best results are indicated in bold.

Table 6. Comparison of nucleotide distribution in the generated sequences for ERD and other methods

Methods	Paired			Unpaired				Total			
	GC	AU	GU	A	C	G	U	A	C	G	U
Natural	0.580	0.293	0.127	0.415	0.161	0.232	0.192	0.250	0.240	0.30	0.210
ERD	**0.690**	0.230	**0.080**	0.476	0.133	**0.243**	0.148	**0.248**	**0.260**	**0.310**	0.182
INFO-RNA	0.950	0.040	0.010	**0.409**	**0.176**	0.261	**0.154**	0.168	0.366	0.400	0.066
MODENA	0.865	0.135	0	0.856	0.056	0.045	0.043	0.410	0.277	0.272	0.041
RNAiFold	0.873	0.106	0.021	0.941	0.016	0.018	0.025	0.444	**0.260**	0.270	0.026
NUPACK	0.723	**0.277**	0	0.377	0.133	0.247	0.243	0.242	0.262	**0.310**	**0.186**

Note: The closest distributions to the natural ones are indicated in bold.

required for successfully designing an RNA sequence for each structure and it is calculated as follows:

$$E_t = \frac{Total\ Execution\ Time}{SC}$$

This simple formula is mathematically equivalent to the expected time appeared in (Garcia-Martin *et al.*, 2013; Taneda, 2011). Because MODENA returns all correctly generated sequences among 50 sequences in its final population as results, the E_t of MODENA is calculated by dividing its execution time by the number of correct solutions to be fair in comparison. As it is mentioned in Tables 4 and 5, our algorithm performs much better than MODENA, RNAiFold and NUPACK in all test samples. Comparing with INFO-RNA, our algorithm is superior especially for longer test samples.

3.3 Nucleotide distribution

The final test we have done over the generated sequences is the nucleotide distribution. We would like to see which method produced RNA sequences with distribution closer to the natural distribution of nucleotides. To do this, we first found the natural distribution of different nucleotides appearing in different structural components. Then we analyzed the generated sequences for each method and extracted the distribution of each nucleotide for different components, as presented in Table 6. As shown in this table, the quality of the sequences generated by our algorithm is much closer to the natural distribution of nucleotides with respect to the other approaches. For example, all mentioned methods (except ERD and NUPACK) use mainly GC base pairs in generating sub-sequences corresponding to the stems. Here, ERD regards the natural distribution of nucleotides comparing with NUPACK. Also, MODENA and RNAiFold use almost always the nucleotide *A* for generating the sub-sequences corresponding to the loops, whereas ERD, INFO-RNA and NUPACK use the other nucleotides as well.

4 CONCLUSION

In this article, a new evolutionary algorithm is proposed for solving the RNA inverse folding problem. Our algorithm is composed of several steps to design a reliable RNA sequence. The results presented in this article indicate that for longer structures, our algorithm performs better than the other mentioned algorithms in terms of accuracy, speedup and nucleotide distribution. Also, the produced energy ranges of our algorithm are comparable with those produced by MODENA. Therefore, our method could generate much more reliable sequences that have diverse stability in terms of energy range and are more similar to the natural RNA sequences.

Although GC base pairs are energetically most favorable, the sequences generated by ERD do not have a high GC content. In the future, it is desirable to introduce sequence constraints in ERD to control the GC content and consequently the stability and the energy range of the designed sequence.

ACKNOWLEDGEMENTS

The authors thank Prof. Peter Clote for his useful comments during this research, Ivan Dotu for helping us to run the RNAiFold program, Hamid Karimi Rouzbahani for editing this manuscript and Saeed Mahmoodi Hashemi and Alireza Shafei Fard for their helpful comments.

Funding: This research was in part supported by a grant from IPM (No. BS-1392-1-06).

Conflict of Interest: none declared.

REFERENCES

Aguirre-Hernández,R. *et al.* (2007) Computational RNA secondary structure design: empirical complexity and improved methods. *BMC Bioinformatics*, **8**, 34.

Andronescu,M. *et al.* (2004) A new algorithm for RNA secondary structure design. *J. Mol. Biol.*, **336**, 607–624.

Avihoo,A. *et al.* (2011) RNAexinv: an extended RNA inverse folding from shape and physical attributes to sequences. *BMC Bioinformatics*, **12**, 319.

Busch,A. and Backofen,R. (2006) INFO-RNA - a fast approach to RNA inverse folding. *Bioinformatics*, **22**, 1823–1831.

Cech,T.R. (2004) RNA finds a simpler way. *Nature*, **428**, 263–264.

Condon,A. *et al.* (2004) Classifying RNA pseudoknotted structures. *Theor. Comp. Sci.*, **320**, 35–50.

Dormi,N. *et al.* (2008) Reconstruction of natural RNA sequences from RNA shape, thermodynamic stability, mutational robustness, and linguistic complexity by evolutionary computation. *J. Biomol. Struct. Dyn.*, **26**, 147–162.

Ganjtabesh,M. and Steyaert,J-M. (2011) Enumerating RNA structures, including pseudoknots of any topology. *MATCH Commun. Math. Comput. Chem.*, **66**, 399–414.

Gao,J-Z.M. *et al.* (2010) Inverse folding of RNA pseudoknot structures. *Algorithms Mol. Biol.*, **5**, 27.

Garcia-Martin,J.A. *et al.* (2013) RNAiFold: a constraint programming algorithm for RNA inverse folding and molecular design. *J. Bioinform. Comput. Biol.*, **11**, 1350001.

Haslinger,C. and Stadler,P. (1999) RNA structures with pseudo-knots: graph-theoretical, combinatorial and statistical properties. *Bull. Math. Biol.*, **61**, 437–467.

Higgs,P.G. (2000) RNA secondary structure: physical and computational aspects. *Q. Rev. Biophys.*, **33**, 199–253.

Hofacker,I.L. *et al.* (1994) Fast folding and comparison of RNA secondary structures. *Monatsh. Chem.*, **125**, 167–188.

Ivry,T. *et al.* (2009) An image processing approach to computing distances between RNA secondary structures dot plots. *Algorithms Mol. Biol*, **4**, 4.

Khan,A.R. *et al.* (2011) NUPACK: analysis and design of nucleic acid systems. *J. Comput. Chem.*, **32**, 170–173.

Knight,J. (2011) Gene regulation: switched on to RNA. *Nature*, **425**, 232–233.

Lyngsø,R.B. *et al.* (2012) Frnakenstein: multiple target inverse RNA folding. *BMC Bioinformatics*, **13**, 260.

Mathews,D.H. *et al.* (1999) Expanded sequence dependence of thermodynamic parameters provides robust prediction of RNA secondary structure. *J. Mol. Biol.*, **288**, 911–940.

McCaskill,J.S. (1990) The equilibrium partition function and base pair probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.

Schnall-Levin,M. *et al.* (2008) Inverting the Viterbi algorithm: an abstract framework for structure design. In: *Proceedings of the 25th International Conference on Machine Learning. Helsinki, Finland, ACM International Conference Proceeding Series*. Vol. 30, pp. 904–911.

Storz,G. (2002) An expanding universe of noncoding RNAs. *Science*, **296**, 1260–1263.

Taneda,A. (2011) MODENA: a multi-objective RNA inverse folding. *Adv. Appl. Bioinform. Chem.*, **4**, 1–12.

Taneda,A. (2012) Multi-objective genetic algorithm for pseudoknotted RNA sequence design. *Front. Genet.*, **3**, 36.

Zadeh,J.N. *et al.* (2011) Nucleic acid sequence design via efficient ensemble defect optimization. *J. Comput. Chem.*, **32**, 439–452.

Zuker,M. (1994) Prediction of RNA secondary structure by energy minimization. *Methods Mol. Biol.*, **25**, 267–294.