A New EA selection function for an efficient RNA design.

(Germany, Leipzig)

Name: Nono and Matteo Revision Number: I
Journal: Date: August 16, 2019

1 Abstract

The inverse RNA folding can be understood as a search problem which consists of searching in a space $S = \{A, C, G, U\}^l$ one of many RNA sequence (s) that fold (s) into a given target structure of length l. Solving this problem is a prior stage for RNA design and is still a real challenge in the field of bioengineering. To improve the RNA inverse folding, we investigate an evolutionary framework implementing a new selection function called Min Ensemble Distance (MED) with a mutation operator which allows the user to fit the nucleotides distribution constrains into the mutation parameters. We implement our algorithm into a simple open source python library called *rnaevol* and the result shows that through this approach, the proposed framework can solve complex targets previously unsolvable by any evolution algorithms on the two main datasets used to benchmark our algorithm by surpassing 7/8 on EteRNA100 dataset and 8/10 on the non-EteRNA dataset. This result demonstrates that taking into account a selection function different to the objective function can significantly improve the performance of an evolutionary algorithm and helps to understand much better the search space.

Key words: RNA design, inverse RNA folding, Evolutionary Algorithm (EA).

2 Introduction and motivation

An RNA molecule is a string over a four letter alphabet, each letter representing a particular nucleotide base: A for adenine, U for uracil, C for cytosine, and G for guanine and it plays an important role in all cellular processes. For an RNA molecule to perform a biological function it has to fold in a three-dimensional structure which strongly depends on its secondary structure. Here we consider the inverse folding problem for which the goal is to find one or many RNA sequences that fold into a given target RNA secondary structure. In another terms it can be considered as an optimization problem where a target RNA secondary structure is given and the goal to determine a set of RNA sequences for which the distance between their corresponding secondary structures and the target one is minimized. In the field of modern bioengineering, to be able to design an RNA molecule which performs a specific function, one must first solve the RNA inverse folding problem [8], [9], [14].

Many methods or algorithms addressing this problem have been proposed in the literature and most of them are performing a stochastic search optimization where an initial potential solutions is generated and refined over a finite number of iterations or generation [5], [4], [6],

[13], [3]. Apart from that there are many other algorithms that are not in the stochastic search basket such as, sentRNA [12] which is a computational agent for RNA design that uses a set of information and strategies collected from the EteRNA game players to train a neural network model that will be enable to predict a set of sequences folding into a given target structure. It is one of the most recent tools. antaRNA [10] utilizes "ant-colony" optimization, in which an initial sequence is generated via a weighted random search, and the goodness of these sequences is then used to refine the weights and improve subsequence over generations. IncaRNAtion generates a GC-weighted partition function for the target structure, and then adaptively samples sequences from it to match a desired GC content. Finally, RNAiFold [7] employs constraint programming that exhaustively searches over all possible sequences compatible with a given target. These approaches have many advantages and disadvantages depending on the technics implemented. NUPACK [15] for example despite its well defined objective function, still has a difficulty of designing sequences for large targets and most of the EteRNA100 targets recently generated by the online RNA game ETERNA. In contrast, ERD [5] because of its strong decomposition method which allows it to deal quickly with large targets (On Rfam 1.0 with target's length between 400-1400) but still a big challenge to solve more than 50% of the ETERNA100 and all other methods are not excluded apart the recent machine learning model sentRNA which was enable to solves 78% of ETERNA100 by adding a refinement on the machine learning model and and without the refinement the deep learning propose by sendRNA can only solve 48% of EteRNA100.

In this work we are more interested in the class of evolutionary algorithms because through its process we can understand much better the RNA sequences (genotype) - (phenotype) RNA structure mapping which can definitively guide our search method. Many works have been done also in the sense of understanding and characterizing the relation between RNA sequence and structure by letting a population of RNA sequence evolve over generation[1] and this can definitively help of designing new evolutionary algorithm to deal much efficiently with the inverse RNA design problem. Amount the methods applying a stochastic search optimization very few of them belong to the class of evolutionary algorithms [5], [6], [13] and from our knowledge up to now none of them is able to solve more than 60% of ETERNA100, for that reason we investigate in this work the choice of a selection function for which the performance of an evolutionary algorithm can be significantly improved and we propose a simple evolutionary algorithm implementing most of the objective functions used by previous tools as a selection function and our proposed selection function. The evolutionary algorihm implements a mutation operation on a population of RNA sequences, the selection is perform by measuring how close to the target structure $\sigma*$ are the structures in the unpseudoknotted structural ensemble Γ of a given sequence s in the population, we call the selection function Min Ensemble Distance (MED). To be able to demonstrate the power of population based algorithm we implemented the Ensemble Defect (ED) selection function (the objective function used by NUPACK) and discuss some of the results obtained. Finally by benchmarking the evolutionary algorithm on Eterna100 we obtain 71% of success and on 87% on the 63 target collected from Rfam [...]. This result actually shows how important is the choice of selection function while designing an evolutionary algorithm for RNA design.

In the first part of our work, we present the general steps of an evolutionary algorithm for inverse RNA folding problem and we explain the main parts of it which are mutation operation and selection method used. In the second part we discuss the choice of evolutionary algorithm parameters: mutation rates and selection function and in the last part we benchmark the method and compared his performance to other methods in the literature.

3 Evolutionary algorithm approaches to solve Inverse RNA folding

3.1 Method

The algorithm that we propose in this work consists of three main parts (See Figure 1):

- Initialization of the population of RNA sequences
- Selection process
- Mutation precess

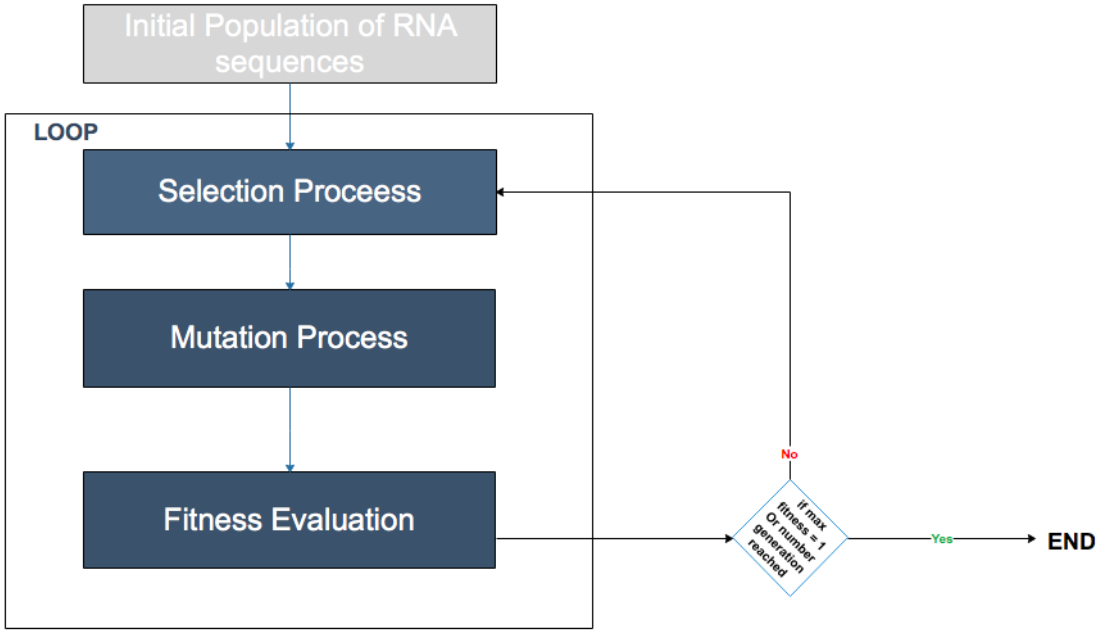


Figure 1: Illustration of an Evolutionary algorithm for RNA design

.

3.1.1 Initialization

For a given population size N and a target structure $\sigma*$ of length l, the initial population of our evolutionary algorithm consists of a population of N RNA sequence generated randomly as follows:

1. Select randomly N nucleotides in a set given by $\{A, U, G, C\}$
2. Identify the base pair position (i, j) in the random sequence previously construct select randomly a base pair in the set of canonic base pairs $\{AU, UA, GU, UG, CG, GC\}$ and fix the first nucleotide of the selected canonic base pair at the position i and the second at position j.
3. repeat 2. for all base par position in the target structure
4. repeat 1. 2. and 3 $N-$times.

3.1.2 Selection methods

As the natural selection is an important step for all evolutionary processes, selection method (or function) plays also an important role within an evolutionary algorithm. In this work we are presenting a new selection method called Min Ensemble distance (MED). and comparing it to three common selection methods used in the previous woks. In our evolutionary algorithm model four selection functions are considered:

- Fitness proportion selection: this is the common selection method used by most of the evolutionary algorithm where the objective function is actually use as a selection criteria to know which individual in the current population will be involved in the next generation. Since the main goal of the inverse folding problem is the find sequences that fold into a given target secondary structure $\sigma*$, the simple fitness measurement f of an RNA sequence ϕ can be defined as follows:

$$f(\phi, \sigma*) = \frac{1}{1 + ham(s^{MFE}(\phi), \sigma*)}$$

  Where ham is the hamming distance between the MFE structure (s^{MFE}) in the ensemble and the target structure ($\sigma*$).

$$s^{MFE}(\phi) = \arg\min_{s \in \Gamma} \Delta G(\phi, s)$$

- Min Ensemble Distance (MED) selection: Here is a new selection function we propose. It is a measure of how close to the target structure $\sigma*$ are the structures in the unpseudo-knotted structural ensemble Γ of a given sequence ϕ in a population. So what the MED selection does is to preserve in the population most of these sequences who have in their structural ensemble the target structure or a structure very close to the target one. The MED selection function measurement is given by:

$$med(\phi, \sigma*) = \min_{s \in \Gamma(\phi)} ham(s, \sigma*)$$

  Where ϕ is an RNA sequence with base identities $\phi_i \in \{A, C, G, U\}$ and $ham(s, \sigma*)$ is the hamming distance between the MFE structure s in in the unpseudoknotted structural ensemble Γ and the target structure ($\sigma*$)

- Ensemble defect (ED) selection [15].

$$n(\phi, \sigma*) = N - \sum_{1<i,j<N} S_{i,j}(s^{MFE}(\phi)) S_{i,j}(\sigma*)$$

  Where $S(s)$ is a structure matrix with entries $S_{i,j} \in \{0, 1\}$. If structure s contains pair $\{i \cdot j\}$, then $S_{i,j}(s) = 1$ otherwise $S_{i,j}(s) = 0$. [ref...]

3.1.3 Mutation operation.

In this work, we present a correlated mutation under two different mutation rates. We define by μ_{bp} the probability of each (i, j) base pair position to be mutated and μ the probability to mutate a non base pair position i in a sequence $(S_i)_{1 \leq i \leq N}$.

Let $N = \{A, U, C, G\}$ be the set of possible nucleotides weighted respectively by a probability:

$$P_N = \{P_A, P_U, P_C, P_G\}$$

And $C = \{AU, UA, CG, GC, UG, GU\}$ be a set of canonic base pairs weighted respectively by a probability:

$$P_C = \{P_{AU}, P_{UA}, P_{CG}, P_{GC}, P_{UG}, P_{UG}\}$$

Where,

$\sum P_N = 1$, $\sum P_C = 1$ and $P_{AU} = P_{UA}$, $P_{UG} = P_{GU}$, $P_{CG} = P_{GC}$

The interesting thing with the mutation performed through the evolution algorithm we are proposing in this work is the flexibility of the mutation rate parameters. Our algorithm allows the user to set the designing constraints explicitly through the mutation parameters P_C and P_N which allows to control the nucleotide distribution during the evolutionary process. The mutation operation is performed as follows:

Algorithm 1: Mutation algorithm

Data: $(S_i)_{1 \leq i \leq N}$, μ, μ_{bp}
Result: Mutated sequence

```
1  i = 1;
2  while i ≤ N do
3      read current;
4      if base pair at (i, j) then
5          select a random uniform number r;
6          if r < μ_bp then
7              select a canonic base pair c ∈ C respect to probability P_C ;
8              S[i] = c[1];
9              S[j] = c[2];
10     else
11         select a random uniform number r;
12         if r < μ then
13             select a nucleotide n_i ∈ N respect to probability P_N ;
14             S[i] = n_i;
15     i = i + 1;
```

An illustration of the execution of the above algorithm is shown on the Figure 2 bellow.

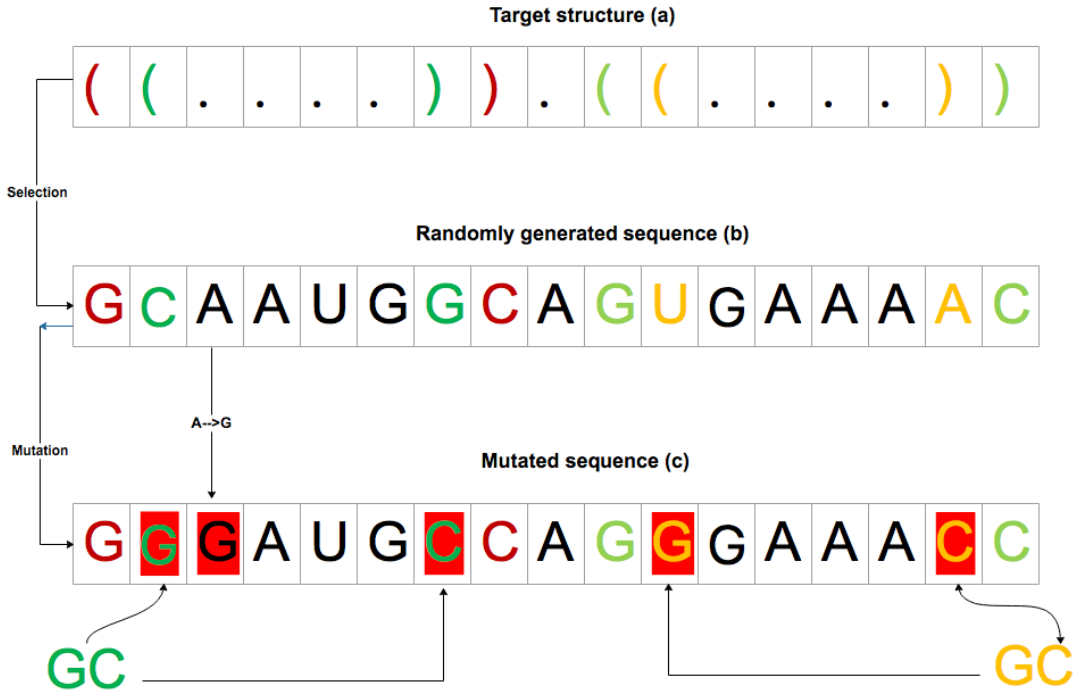


Figure 2: Illustration of mutation process

(a) is a given target structure and **(b)** is a random generated sequence selected from an N-population of RNA-sequences. **(c)** is the mutated sequence where the non-base pair positions (the part in black) are mutated with a probability $\mu = 1/17$, so only one nucleotide has been mutated at the third position $(A - - > G)$. The base pair positions are mutated with a probability $\mu_{bps} > \mu$ (the coloured part above) four positions have been mutated $(1, 6)$ and $(11, 16)$.

To summarize the above steps, the evolutionary algorithm implementing the selection function we propose is given by:

Algorithm 2: rnaevol algorithm

Data: N, max_{gen}, μ, μ_{bp}
Result: nextpopulation

$n = o$;
prevpopulation = initipop(N);
max_fitness = max(prevpopulation);
while $n \leq max_{gen}$ & $max\_fitness < 1.0$ **do**
    nextpopulation = copy of the 10 best individuals in prevpopulation;
    evaluate(prevpopulation);
    selected_ind = select(prevpopulation, size, method);
    for $ind \in selected\_ind$ **do**
        mutated_ind.add(mutate(ind, μ, μ_{bp}))
    nextpopulation = nextpopulation.add(mutated);
    prevpopulation = copy(nextpopulation);
    max_fitness = max(prevpopulation);
    $n = n + 1$;

4 Results and discussions

In this section we would like to address two main parts of our work. First the choice of the evolutionary algorithm parameters, secondly we discuss the importance of the selection function and finally, using two different benchmark data, we compare the results obtained and the performance of our method to the existent one. In all the experiments carried out in this work, we are using the python library *rnaevol* (implementing our proposed EA framework) which depends on ViennaRNA Package 2.0 [11] including: RNAfold, RNAsubopt, RNAeval, etc...

4.1 Parameters

One of the main challenge for an Evolutionary algorithm to perform good result is to find right parameters that fit into the algorithm. The EA proposed in this work has a set of parameters that need to be tuned. We have:

- The population size N and the number of generation max_{gen}: The is always a trade-off between the number of generation and the population size needed by an EA to perform efficiently. In most of the cases less the population size is more we need generation and vice versa. For the matter of resources and to evaluate our algorithm in the worst case the population size we have chosen doest not exceed 120 individuals peer generation. in the most case $N = 100$. and $max_{gen} < 500$.
- The mutation parameters μ, μ_{bp}, P_N and P_C: For all the experiment we perform in this work μ is chosen proportionally to the length of the target generally $\frac{1}{\text{lenght of the target}}$ and $\mu_{bp} >> \mu$ for most cases $\mu_{bp} = 0.5$ to allow half of the base pairs positions of a sequence to be mutated. the parameters P_C and P_N are chosen based on the nucleotide distribution in natural RNA in [5].
- The selection function: For the selection function, the most used is MED function and something switch to the fitness selection function due to the computation time which in our case strongly depends on the length of the target.

4.2 EA's selection methods analysis

As mentioned earlier the mains challenge in this work is to choose which of the three selection method is the best for an evolutionary algorithm to solve the inverse RNA folding problem. To compare them we have chosen some target structures from the ETERNA database on which we have performed the following analysis:

- Compare the mean fitnesses over generation over generation.
- Counting the number of success over all the experiments performed
- For each success, reconstruct the genealogy of the structures and compare their fitness and the mean fitness at each generation.

To be able to compare the three selection methods the first target structure chosen from ENTERNA database is Shortie 6 for two reasons, first because of the short structure length of 35 which allows us to repeat the experiment many time and secondly because was one of the structure difficult to design and used by Jeff and al in [2] to compare RNA-design tools such as NUPACK, INFIO-RNA, RNA-Inverse, RNA-SSD, etc...

The Table bellow summarizes the result.

Table 1: Comparison of selection methods on Short 6

Selection methods	Ens Def.	Min Ens Dist.	Fitness
Average number of generations	229.84	280.59	500
Median number of generations	192.	236	500
Number od success	90	67	0

The result above is obtained by repeating 100 times, for a maximum of 500 generations, the mutation probabilities $\mu = \frac{1}{35}$ and $\mu_{bp} = 0.5$ without elitism and using an initial population generated using the steps above section 3.1.1. The Table 1 shows that the ED selection method solves with a success rate of 90% and the MED follows with a success rate of 67%.

The surprising observation that we can make here is that in the paper of Jeff et al. [2] where the authors compared the existing tools by benchmarking them on Eterna100 the tool NUPACK which solves the inverse folding by minimizing the ensemble defect did not solve this structure, so which means that through an evolutionary algorithm where the selection function minimizes the ensemble defect can perform better.

The Figure 3 **(a)** bellow show that for one of the success run the ensemble defect of the sequence found is actually less than the one obtained by running the same structure on the last version of NUPACK web server. The sequence found by NUPACK web server is given by:

$$GGAAUACCAGAGAGAUCAGCGUUUGCACCACCUGG$$

and using the RNAfold tool from ViennaRNA package the sequence folds into the following structure:

$$.....((((.(.....((....)).....))))))(-3.10)$$

And the corresponding ensemble defect is 15.58 which is higher that the one obtained by our method which is under 8.0. as we can observe on Figure 3 **(a)**.

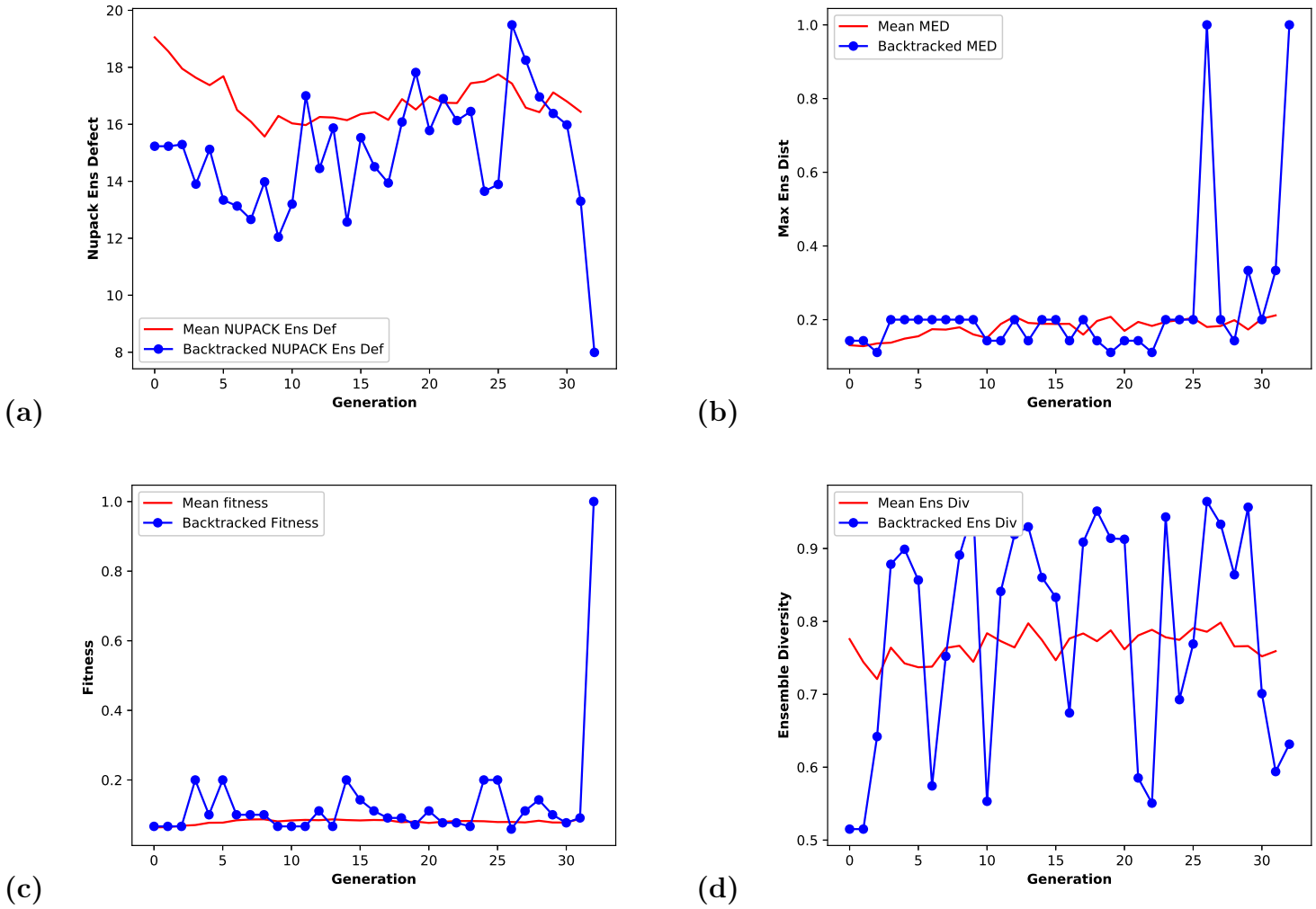
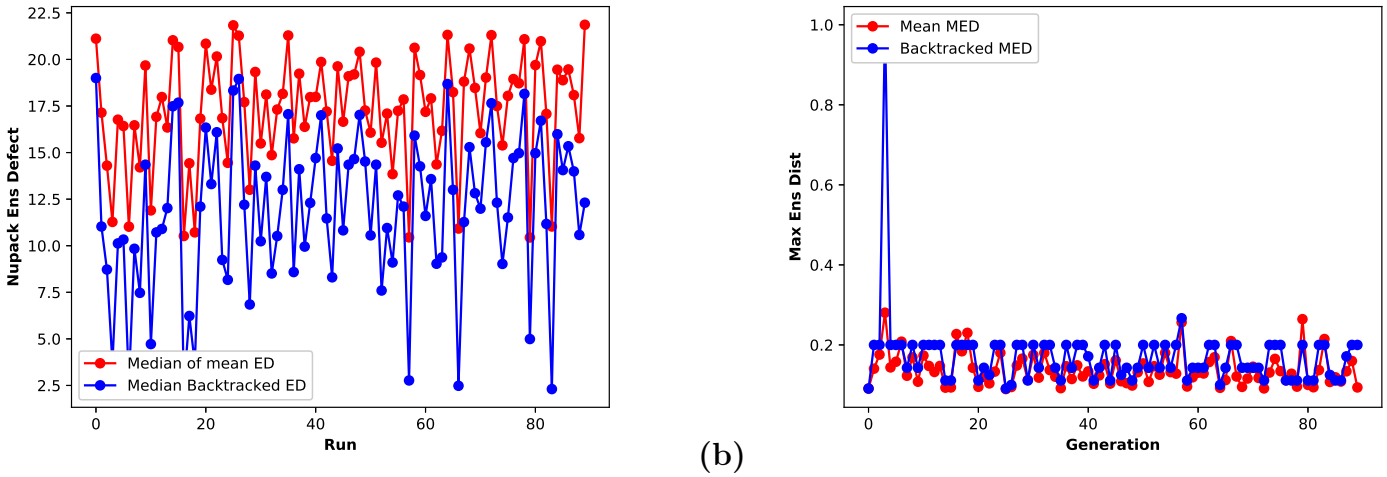


Figure 3: Shortie 6 structure analysis.

(a) is showing the mean ensemble defect of a population od 100 sequences over generation compared to the ensemble defect of all sequence in the genealogy of the found structure, **(b)** the min ensemble distance, **(c)** the fitness and the last plot **(d)** shows the means ensemble diversity compared to the ensemble diversity of backtracked sequences. As we can observe for this run that the ensemble defect and the min ensemble distance selection methods seems to be promising. Firstly because on **(a)** and **(b)** most of the backtracked sequences ensemble defects computed are bellow the mean ensemble and the same for the min ensemble distance they are above. Further more, if we look at the last sequences before we hit the target structure they are bellow the mean for the ED and above for the MED. In contrast on **(d)** is not the same case for the ensemble diversity which was expected to be above the mean too.

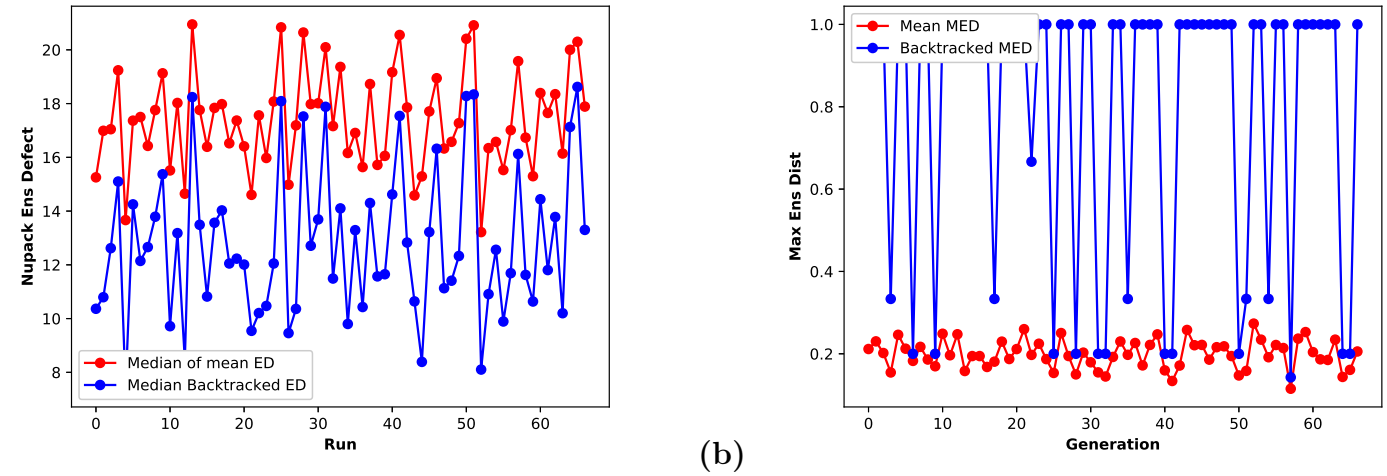
On Figure 4 bellow we have a plot summarizing the 90 success runs for ED selection and on Figure 5 the 67 success runs for MED selection method.



(a) (b)

Figure 4: Shortie 6 structure analysis for the 90 success run of ED's selection method.

(a) is showing for each success run in blue the median of the distribution of backtracked sequences ensemble defect and in red the median of the distribution of mean ensemble defect in a population of 100 sequences over generation and on **(b)** the same observation for the min ensemble distance.



(a) (b)

Figure 5: Shortie 6 structure analysis for the 67 success run of MED's selection

(a) is showing for each success run in blue the median of the distribution of backtracked sequences ensemble defect and in red the median of the distribution of mean ensemble defect in a population of 100 sequences over generation and on **(b)** the same observation for the min ensemble distance

As explained earlier for one MED success run, we can make the same observation for all the success runs of each method on Figure 4 and Figure 5 which show in both cases that for the majority of success runs the median of ensemble defect and the min ensemble distance of backtracked sequences are respectively bellow the median of the mean distribution of ED in the population and above the median of the mean ensemble.

We repeated the same experiment above on a different target structure chosen from ENTERNA database called *Small and Easy 6* for the same reasons as the first one.

The Table bellow summarizes the result.

Table 2: Comparison of selection methods on Small and Easy 6

Selection methods	Ens Def.	Min Ens Dist.	Fitness
Average number of generations	486.48	456.97	450.04
Median number of generations	500	500	500
Number od success	6	17	14

The result above is obtained by repeating 100 times, for a maximum of 500 generations, the mutation probabilities $\mu = \frac{1}{30}$ and $\mu_{bp} = 0.5$ without elitism and using an initial population generated using the steps above section 3.1.1. As we can notice in Table 2 the success rate for this second target structure is much lower than the preview one. The ED's selection method is the least one with 6% of success. But it is also important to notice that after 100 runs using NUPACK tool we did not get success and the best sequence designed by NUPACK is:

$$AAUGCGUCAUGCUAUCCAUCUGCGGGAAUU$$

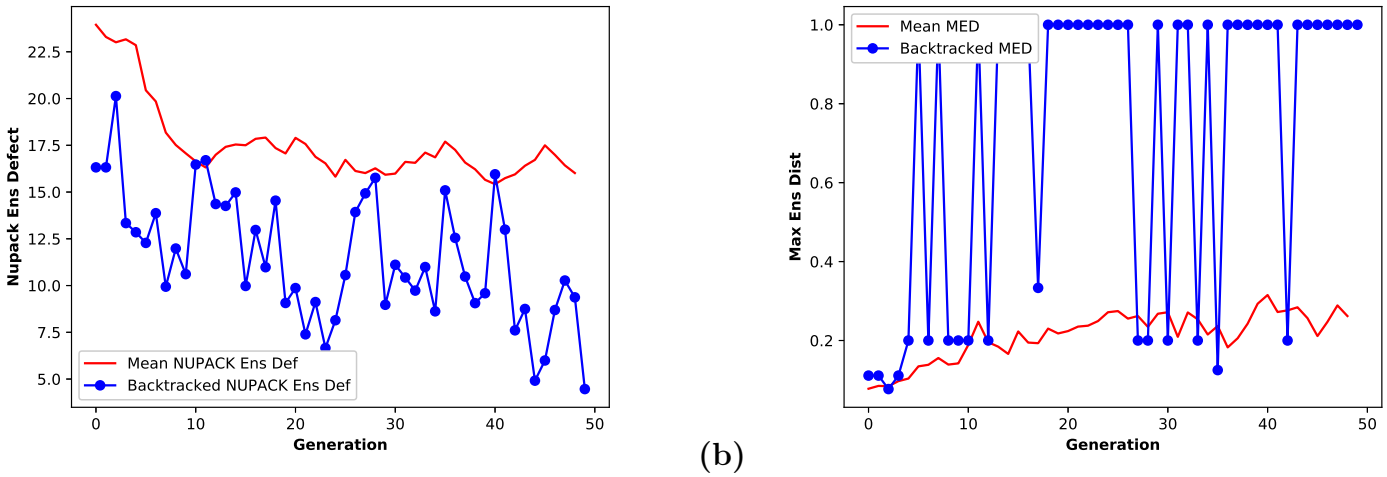
and using the RNAfold tool from ViennaRNA package the sequence folds into the following structure:

$$.....((.(((.........))).))...(-2.00)$$

And the corresponding ensemble defect is 9.75 which is higher that the one obtained by our method which is bellow 5.0 as we can see on the Figure 6 **(a)**.

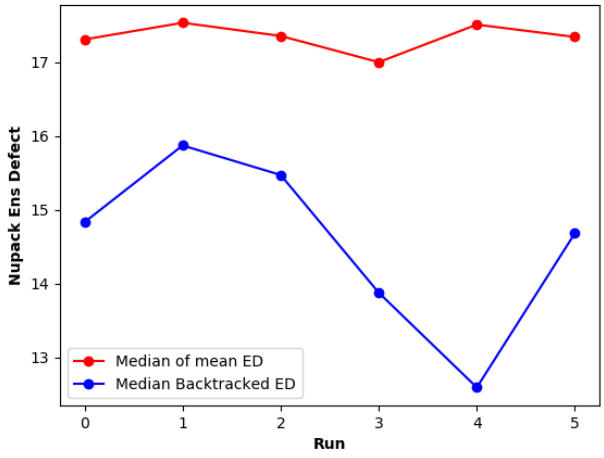
Performing the same analysis as previously, we can notice from Figure 6 the same behaviour for one of the success run. Despite of low success rate of ED's method the ensemble defect selection method seems to be more promising that MED because the ensemble defect of all the backtracked sequences is significantly bellow the mean ensemble defect of the population which shows that maintaining a population of sequences with low ensemble defect will definitely help to hit the the sequence folding into the target structure. It also important to say that it is almost the same case for the MED selection method.

The plots summarizing the 6 success runs of ED's method and the 17 are also shown bellow on Figure 8 and Figure 7.

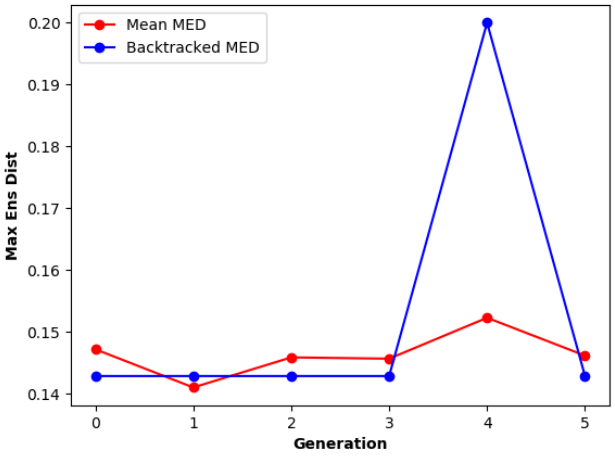


(a) **(b)**

Figure 6: Small and Easy 6 structure analysis.



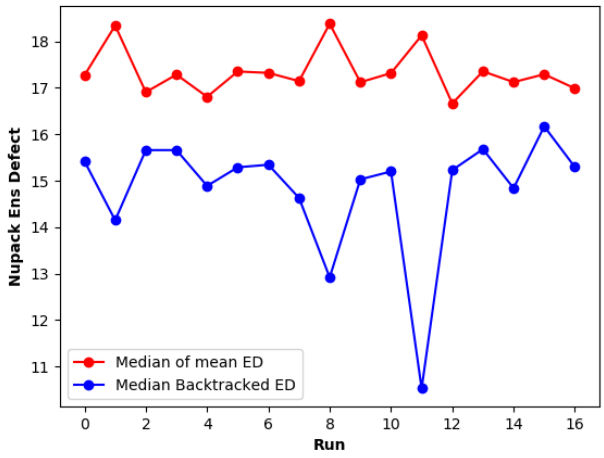
(a)



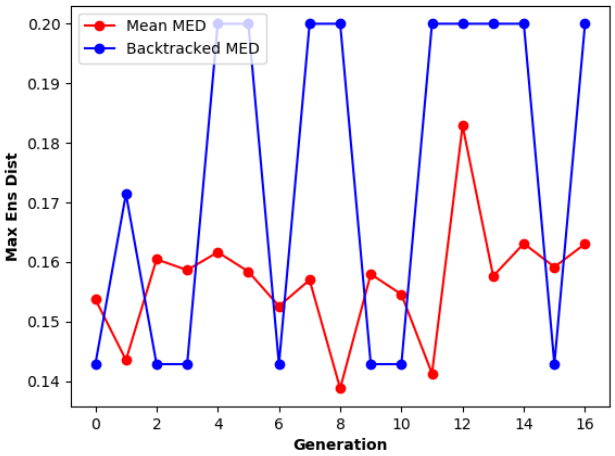
(b)

Figure 7: Small and Easy 6 structure analysis for the 9 success run of ED's selection method.

(a) is showing for each success run in blue the median of the distribution of backtracked sequences ensemble defect and in red the median of the distribution of mean ensemble defect in a population of 100 sequences over generation and on **(b)** the same observation for the min ensemble distance. On **(b)** we can notice that in most case the median of the backtracked sequences is not greater than the median of the mean distribution which makes sense because the selection pressure was on ensemble defect.



(a)



(b)

Figure 8: Small and Easy 6 structure analysis for the 17 success run of MED's selection

(a) is showing for each success run in blue the median of the distribution of backtracked sequences ensemble defect and in red the median of the distribution of mean ensemble defect in a population of 100 sequences over generation and on **(b)** the same observation for the min ensemble distance. In contrast of the previous one, on **(a)** we can notice that in most case the median of ED backtracked sequences is not less than the median of the mean ED distribution which means that some how selecting on MED is also minimizing the ED.

4.3 Benchmark

As we can notice from the previous preformed experiments , we can not really decide which of the selection methods perform better or is well suited for an EA to deal on inverse folding because in a Shortie 6 the ensemble defect perform better and in a Small and short 6 the MED performs better. So it is important to benchmark them using the entire dataset and compare their result to existing methods.

4.3.1 Data description

To evaluate and compare the performance of the EA proposed in this work to other existing methods in the literature we have have chosen to use the two main datasets used by the recent tool deep learning tool SentRNA [12]:

- The Eterna100 which contains a set of 100 target structures extracted from Eterna Pulzze game players and classifier by their degree of difficulty.
- A set of 63 non-Eterna experimentally synthesized targets that Garcia-Martin et al. recently used to benchmark a set of 10 inverse folding algorithms which for our knowledge, this is the most recent and comprehensive benchmark of current state-of-the-art methods as also mentioned in [12]. The dataset is collected from 3 sources: the first dataset called **dataset A** which contains 29 targets [5] [13] and the second called **dataset B** is a collection of 24 targets used in [5] and added to that the 10 used in [12].

4.3.2 First observations

In this first observation, we compare our EA's performance on the Eterna100 benchmark dataset. The result in Table 3 bellow is obtained by running in parallel for each target structure $\sigma*$ in the EteRNA100 4 instances of our EA, for a maximum of 200 generations under a mutation rate of $\mu = \frac{1}{lenght(\sigma*)}$ and $\mu_{bp} = 0.5$ with elitism and using an initial population of 100 RNA sequences generated using the above steps in section 3.1.1. In the following table, "*rnaevol + GC pairing*" corresponds to the mutation parameter $P_N = \big[P_A = 0.3, P_U = 0., P_C = 0., P_G = 0.7\big]$ and $P_C = \big[P_{AU} = P_{UA} = 0, P_{CG} = P_{GC} = 0.5, P_{UG} = P_{UG} = 0\big]$, for the case "*rnaevol + All pairing*" all the nucleotides have the same chance of being selected and the base pairs as well.

As we can notice in the Table 3 the evolution algorithm we propose, can solve 71% using MED's selection method and 64% using ED's selection method which is sufficient to surpass 6/7 methods benchmarked by ER et al. in [12]. The performances of an EA using the MED's selection method with all pairing and the one using only GC pairing are almost the same Using an ED's selection + GC, we can actually solve 23 targets more than NUPACK which is also minimizing the ensemble defect and that shows the importance of population based algorithm. Our proposed EA with a new selection function MEA can solve 17 target more than the previous GA-based algorithms MODENA [13] [6].

Table 3: Summary of performance of rnaevol vs the 7 other algorithms benchmarked on the Eterna100 by Anderson-Lee et al.

Methods	Number of puzzles solved
rnaevol with MED selection + GC pairing	**71/100**
rnaevol with MED selection + All pairing	60/100
rnaevol with ED selection + GC pairing	**67/100**
rnaevol with ED selection + All pairing	50/100
SentRNA, NN only	47/100
SentRNA, NN + full moveset	78/100
SentRNA, NN + GC pairing	74/100
SentRNA, NN + All pairing	72/100
RNAinverse	28/100
RNA-SSD	27/100
DSS-Opt	47/100
NUPACK	48/100
INFO-RNA	50/100
MODENA	54/100

4.3.3 Second observations

For this second experiment carried out we use the second dataset of 63 targets structures which is a combination of dataset A and dataset B. The statical result compared to others tools is presented in Table 4.

We use the same parameters like in the first experiment apart from the number of generation which is reduce to 100 and the mutation parameters P_C and P_N. P_C and P_N were chosen to be close to the nucleotide distribution of the RNA sequence in the nature [5]. The result shows that our method surpasses 8/10 other methods. ERD is solving 2 targets more than our method because as we stated earlier one of the main advantage of ERD is its strong decomposition capacity which allows it to solve the entire dataset B. For the reason that the targets are very long in the dataset B we used the fitness selection method which allows us to speed up and with the advantage that our evolutionary algorithm also allows us to fit the nucleotide distribution parameters taken from natural RNA directly in the mutation parameters, we are able to solve 21/24 targets from this **dataset B**. For the **dataset A** we use the MED selection method and we solve 24/29 target which means 2 more than the previous tools and for the 10 last targets we solve 7/10. Adding all this solved target together we obtain a result of 52/63 presented in Table 4

Table 4: Summary of performance of rnaevol vs the 10 other algorithms benchmarked on the non-Eterna100 by Anderson-Lee et al.

Methods	Number of puzzles solved
RNAEVOL	**52/63**
SentRNA, NN only	46/63
SentRNA, NN + full moveset	57/63
SentRNA, NN + GC pairing	53/63
SentRNA, NN + All pairing	53/63
RNAfbinv	0/63
IncaRNAtion	28/63
Frnakenstein	27/63
RNAinverse	20/63
RNA-SSD	47/63
NUPACK	29/63
INFO-RNA	45/63
MODENA	32/63
ERD	54/63

4.3.4 Accuracy and speed comparisons on the dataset A

After benchmarking our algorithm on existent datasets and compare the result obtained to previous tools, we also found important to address the accuracy and the speed comparisons of our methods. Here we present time ans accuracy comparison on the **dataset A**. The formula used to compute $E_t(s)$ is the same used in [5] and it is given by:

$$E_t(s) = \frac{\text{Total Excution Time}}{Sc}$$

And it is important to say that we did not run the experiment on all other tools we took the result performed in [5].

In Table, the best results are indicated in bold, as we can see the 2 targets: **RF00028** and **RF00028** that our algorithm was able to solve more than INFO-RNA and MODENA and the 2 targets: **RF00030** and **RF00018** more than ERD. This allows our algorithm to obtain a best accuracy over all other existing algorithm found in the literature. But we can also notice that even though for a very few cases we have gotten better speed, the computational time is much more expensive than ERD and INFO-RNA and this can be understood by the fact that when we use a selection function different to the fitness function we evaluate every individual tweest at each generation and for a long target we have an exponential growth of the unpseudoknotted structural ensemble Γ of a given sequence in the population and an energy range which consequently increase exponentially the computation time of our algorithm. To avoid an infinite time computation for long targets we set the the maximum energy of each structure in the ensemble Γ up to 3 $Kcal.mol^{-}1$.

Table 5: Summary of performance of *rnaevol* vs the 4 other algorithms benchmarked on the **dataset B** by A.Esamaili-Taheri et al

ID	Length(nt)	rnaevol		ERD		INFO-RNA		MODENA		NUPACK	
		$E_t(s)$	SC	$E_t(s)$	SC	$E_t(s)$	SC	$E_t(s)$	SC	$E_t(s)$	SC
RF00008	54	0.115	50	0.16	50	0.011	50	41.905	49	7.005	50
RF00029	73	0.371	50	0.373	50	0.033	50	71.502	38	41.415	41
RF00005	74	0.227	50	0.045	50	0.032	50	36.046	47	24.85	50
RF00027	79	0.2088	50	0.1	50	0.119	50	61.825	40	20.732	50
RF00019	83	0.336	50	0.136	50	0.091	50	50.813	35	9.211	50
RF00014	87	0.2914	50	0.103	50	0.034	50	63.707	35	6.389	50
RF00006	89	0.4	50	0.23	50	0.255	50	37.926	41	51.291	50
RF00026	102	0.36	50	0.038	50	11.294	50	57.124	44	209.624	50
RF00001	117	1,51	50	2.389	49	0.255	50	53.41	46	25423.278	1
RF00021	118	0.556	50	0.3	50	0.164	50	59.139	46	18.411	50
RF00020	119	∞	0	∞	0	∞	0	∞	0	∞	0
RF00016	129	∞	0	∞	0	∞	0	∞	0	4028.944	7
RF00015	140	2.46	50	2.218	49	2.885	50	77.096	42	∞	0
RF00022	148	**2.01**	**50**	2.846	50	6.384	50	77.024	44	1870.129	14
RF00002	151	**3.76**	**50**	7.831	49	11.294	50	64.602	39	∞	0
RF00007	154	2.64	50	1.429	50	0.975	50	85.534	43	531.148	32
RF00003	161	34,62	34	23.325	43	379.851	14	∞	0	∞	0
RF00013	185	5.92	50	3.014	50	8.601	50	104.908	46	18.1	50
RF00004	193	6,24	50	3.177	50	5.883	50	84.767	41	814.047	14
RF00025	210	11,34	50	7.584	50	7.361	50	106.894	48	∞	0
RF00012	215	13.8	50	3.164	50	44.89	49	111.521	48	345.226	46
RF00017	301	144	50	4.331	50	1.373	50	350.1	48	69.3605	50
RF00030	340	(1125/10)	50	∞	0	207.593	50	294.552	36	∞	0
RF00028	344	(-/8.7)	50	147.091	44	∞	0	∞	0	∞	0
RF00009	348	(-/10)	50	268.613	28	∞	0	288.867	45	∞	0
RF00010	357	∞	0	∞	0	∞	0	∞	0	∞	0
RF00018	360	(-/34.37)	38	∞	0	296.386	50	293.777	39	∞	0
RF00011	382	∞	0	∞	0	∞	0	∞	∞	0	
RF00024	451	∞	0	∞	0	∞	0	∞	0	1387.425	12
SUM	-	(-/305.6852)	**1172**	478.367	1062	986.298	1046	2467.039	940	34876.594	682

5 Conclusion and perspective

In this work, we investigated the place of a selection method within an evolutionary algorithm, and as a discovery, we proposed a simple evolutionary approach implementing the new selection method for solving the RNA inverse folding problem. The result shows that our evolutionary algorithm based on MED/fitness selection performs better than the existing evolutionary algorithms in the literature and surpass 6/7 methods benchmarked by ER et al.[12] on the Eterna100 benchmark first and secondly in the dataset of 29 biological target structures used to measure the algorithm's performance of previous

existing algorithms. Therefore, it will be important to take into account the choice of selection function independently of the objective function to be able to generate much more reliable and efficient solutions of the inverse RNA folding.

References

[1] L. W. Ancel and W. Fontana. Plasticity, evolvability, and modularity in rna. *Journal of Experimental Zoology*, 288(3):242–283, 2000.

[2] J. Anderson-Lee, E. Fisker, V. Kosaraju, M. Wu, J. Kong, J. Lee, M. Lee, M. Zada, A. Treuille, and R. Das. Principles for predicting rna secondary structure design difficulty. *Journal of molecular biology*, 428(5):748–757, 2016.

[3] A. Churkin, M. D. Retwitzer, V. Reinharz, Y. Ponty, J. Waldispühl, and D. Barash. Design of rnas: comparing programs for inverse rna folding. *Briefings in bioinformatics*, 19(2):350–358, 2017.

[4] N. Dromi, A. Avihoo, and D. Barash. Reconstruction of natural rna sequences from rna shape, thermodynamic stability, mutational robustness, and linguistic complexity by evolutionary computation. *Journal of Biomolecular Structure and Dynamics*, 26(1):147–161, 2008.

[5] A. Esmaili-Taheri and M. Ganjtabesh. Erd: a fast and reliable tool for rna design including constraints. *BMC bioinformatics*, 16(1):20, 2015.

[6] A. Esmaili-Taheri, M. Ganjtabesh, and M. Mohammad-Noori. Evolutionary solution for the rna design problem. *Bioinformatics*, 30(9):1250–1258, 2014.

[7] J. A. Garcia-Martin, P. Clote, and I. Dotu. Rnaifold: a constraint programming algorithm for rna inverse folding and molecular design. *Journal of bioinformatics and computational biology*, 11(02):1350001, 2013.

[8] M. S. Goldberg, D. Xing, Y. Ren, S. Orsulic, S. N. Bhatia, and P. A. Sharp. Nanoparticle-mediated delivery of sirna targeting parp1 extends survival of mice bearing tumors derived from brca1-deficient ovarian cancer cells. *Proceedings of the National Academy of Sciences*, 108(2):745–750, 2011.

[9] C. Hao, X. Li, C. Tian, W. Jiang, G. Wang, and C. Mao. Construction of rna nanocages by re-engineering the packaging rna of phi29 bacteriophage. *Nature communications*, 5:3890, 2014.

[10] R. Kleinkauf, M. Mann, and R. Backofen. antarna: ant colony-based rna sequence design. *Bioinformatics*, 31(19):3114–3121, 2015.

[11] R. Lorenz, S. H. Bernhart, C. H. Zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. Viennarna package 2.0. *Algorithms for molecular biology*, 6(1):26, 2011.

[12] J. Shi, R. Das, and V. S. Pande. Sentrna: Improving computational rna design by incorporating a prior of human design strategies. *arXiv preprint arXiv:1803.03146*, 2018.

[13] A. Taneda. Modena: a multi-objective rna inverse folding. *Advances and applications in bioinformatics and chemistry: AABC*, 4:1, 2011.

[14] M. N. Win and C. D. Smolke. Higher-order cellular information processing with synthetic rna devices. *Science*, 322(5900):456–460, 2008.

[15] J. N. Zadeh, B. R. Wolfe, and N. A. Pierce. Nucleic acid sequence design via efficient ensemble defect optimization. *Journal of computational chemistry*, 32(3):439–452, 2011.