A weighted sampling algorithm for the design of RNA sequences with targeted secondary structure and nucleotides distribution

Vladimir Reinharz, Yann Ponty, Jérôme Waldispühl

BIOINFORMATICS

Vol. 00 no. 00 2013
Pages 1–9

A weighted sampling algorithm for the design of RNA sequences with targeted secondary structure and nucleotides distribution

Vladimir Reinharz[1], Yann Ponty[2,*], Jérôme Waldispühl[1*]

[1] School of Computer Science, McGill University, Montreal, Canada
[2] Laboratoire d'informatique, École Polytechnique, Palaiseau, France.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivations: The design of RNA sequences folding into predefined secondary structures is a milestone for many synthetic biology and gene therapy studies. Most of the current software uses similar local search strategies (i.e. a random seed is progressively adapted to acquire the desired folding properties) and more importantly do not allow the user to control explicitly the nucleotide distribution such as the GC-content in their sequences. However, the latter is an important criterion for large-scale applications as it could presumably be used to design sequences with better transcription rates and/or structural plasticity.

Results: In this paper, we introduce `IncaRNAtion`, a novel algorithm to design RNA sequences folding into target secondary structures with a predefined nucleotide distribution. `IncaRNAtion` uses a global sampling approach and weighted sampling techniques. We show that our approach is fast (i.e. running time comparable or better than local search methods), seed-less (we remove the bias of the seed in local search heuristics), and successfully generates high-quality sequences (i.e. thermodynamically stable) for any GC-content. To complete this study, we develop an hybrid method combining our global sampling approach with local search strategies. Remarkably, our glocal methodology overcomes both local and global approaches for sampling sequences with a specific GC content and target structure.

Availability: `IncaRNAtion` is available at csb.cs.mcgill.ca/incarnation/

Contact: jeromew@cs.mcgill.ca, yann.ponty@lix.polytechnique.fr

Key words: RNA, secondary structure, design, weighted sampling, GC-content.

1 INTRODUCTION

At the core of the emerging field of synthetic biology resides our capacity to design and re-engineer molecules with target functions. RNA molecules are well tailored for such applications. The ease to synthesize them (they are directly transcribed from DNA) and the broad diversity of catalytic and regulation functions they can perform enable to integrate *de-novo* logic circuits within living cells (Rodrigo *et al.*, 2012) or re-program existing regulation mechanisms (Chang *et al.*, 2012). Future advances and applications of these techniques in gene-therapy studies will strongly rely on efficient computational methods to design and re-engineer RNA molecules.

Most of RNA functions are, at least partially, encoded by the three-dimensional molecular structures, which are themselves primarily determined by the secondary structures. The development of efficient algorithms for designing RNA sequences with pre-defined secondary structures is thus a milestone to enter the synthetic biology era. `RNAinverse` pioneered RNA secondary structure design algorithms. It has been developed and distributed with the Vienna RNA package (Hofacker *et al.*, 1994). However, only posterior experimental studies revealed the potential and practical impact of these techniques. Thereby, during the last 6 years many improvements and variants of `RNAinverse` have been proposed. Conceptually, almost all of existing algorithms follow the same approach. First a seed sequence is selected, then a local search strategy is used to mutate the seed and find, in its vicinity, a sequence with desired folding properties. Using this strategy, `INFO-RNA` (Busch and Backofen, 2006), `RNA-SSD` (Aguirre-Hernández *et al.*, 2007) and `NUPACK:Design` (Zadeh *et al.*, 2011) significantly improved the performance of RNA secondary structure design algorithms. More recent research studies aimed to include more constraints in the selection criteria. `RNAexinv` focused on the design of sequences with enhanced thermodynamical and mutational robustness (Avihoo *et al.*, 2011), while `Frnakenstein` enables to design RNA with multiple target structures (Lyngsø *et al.*, 2012).

We recently introduced with `RNA-ensign` a novel paradigm for the search strategy of RNA secondary structure design algorithm (Levin *et al.*, 2012). Instead of a local search approach, we proposed a global sampling strategy of the mutational landscape based on the `RNAmutants` algorithm (Waldispühl *et al.*, 2008). This methodology offered promising performances, but suffered from prohibitive runtime and memory consumption. Following our work, Garcia-Martin *et al* proposed `RNAiFOLD` (Garcia-Martin *et al.*, 2013), an alternate methodology that uses constraint programming techniques to prune the mutational landscape. While also suffering from prohibitive running times, it is worth noting that this latter algorithm also proposes a seed-less approach to the RNA secondary structure design problem.

In this paper, we introduce `IncaRNAtion`, a RNA secondary structure design algorithm that benefits of our recent algorithmic advances (Reinharz *et al.*, 2013) to expand our original

*to whom correspondence should be addressed

`RNA-ensign` algorithm (Levin *et al.*, 2012). `IncaRNAtion` addresses previous limitations of `RNA-ensign` and offers new functionalities. First, while our previous program had a running time complexity of $\mathcal{O}(n^5)$, `IncaRNAtion` now runs in linear-time and space complexity, allowing it to demonstrate similar speeds as any local search algorithm. Next, `IncaRNAtion` is *seed-less*. Unlike `RNA-ensign`, it does not require a seed sequence to initiate its search. Finally, `IncaRNAtion` implements a novel algorithm based on weighted sampling techniques (Bodini and Ponty, 2010) that enables us to control, for the first time, *explicitly* the GC-content of the solution. This functionality is essential because wild-type sequences within living organisms often present medium or low GC-content, presumably to offer better transcription rates and/or structural plasticity. Previous programs do not allow to control this parameter and tend to output sequences having high GC-contents (Lyngsø *et al.*, 2012).

We demonstrate the performance of our algorithms on a set of real RNA structures extracted from the RNA STRAND database (Andronescu *et al.*, 2008). To complete this study, we develop an hybrid method combining our global sampling approach with local search strategies such as the one implemented in `RNAinverse`. Remarkably, our glocal methodology overcomes both local and global approaches for sampling sequences with a specific GC content and target structure.

2 METHODS

We introduce a probabilistic model for the design of RNA sequences with a specific GC-content and folding into a predefined secondary structure. For the sake of simplicity, we choose to base this proof-of-concept implementation on a simplified free-energy function $E(\cdot)$, which only considers the contributions of stacked canonical base-pairs. We show how a modification of the dynamic programming scheme used in `RNAmutants` allows for the sampling of good and diverse design candidates, in linear time and space complexities.

2.1 Definitions

A targeted secondary structure S^* of length n is given as a non-crossing arc-annotated sequence, where S^*_i stands for the base-pairing position of position i in S^* if any (and, reciprocally, $S^*_{S^*_i} = i$), or -1 otherwise. In addition, let us denote by $\#\mathsf{gc}(s)$ the number of occurrences of G and C in an RNA sequence s.

2.1.1 Simplified energy model We use a simplified free-energy model which only includes additive contributions from stacking base-pairs. Using individual values from the Turner 2004 model (retrieved from the NNDB (Turner and Mathews, 2010)). Given a candidate sequence s for a secondary structure S, the free-energy of any sequence s of length $|S|$ is given by

$$E(s,S) = \sum_{\substack{(i,j)\to(i',j')\in S \\ \text{stacking pairs}}} E^{\beta}_{s_i s_j \to s_{i'} s_{j'}}$$

where $E^{\beta}_{ab\to a'b'}$ is set to 0 if $ab = \varnothing$ (no base-pair to stack onto), the tabulated free-energy of stacking pairs $(ab)/(a'b')$ in the Turner model if available, or $\beta \in [0,\infty]$ for non-Watson-Crick/Wobble pairs (i.e. not in $\{\mathsf{GU},\mathsf{UG},\mathsf{CG},\mathsf{GC},\mathsf{AU}$ or $\mathsf{UA}\}$). This latter parameter allows one to choose whether to simply penalize invalid base pairs ($\beta > 0$), or forbid them altogether ($\beta = +\infty$). Position-specific sequence constraints can also be enforced at this level (details omitted for the sake of clarity) by assigning to E a $+\infty$ penalty (leading to a null probability) in the presence of a base incompatible with a user-specified constraint mask.

2.1.2 GC-weighted Boltzmann ensemble and distribution In order to counterbalance the documented tendency of sampling methods to generate GC-rich sequences (Levin *et al.*, 2012), we introduce a parameter $x \in \mathbb{R}^+$, whose value will influence the GC-content of generated sequences. For any secondary structure S, the GC-weighted-Boltzmann factor of a sequence s is $\mathcal{B}^{[x]}_S(s)$ such that

$$\mathcal{B}^{[x]}_S(s) = e^{\frac{-E(s,S)}{RT}} \cdot x^{\#\mathsf{gc}(s)} \tag{1}$$

where R is the Boltzmann constant and T the temperature in Kelvin.

Summing the GC-weighted-Boltzmann factor over all possible sequences of a given length $|S|$, one obtains the GC-weighted partition function $\mathcal{Z}^{[x]}_S$, from which one defines the GC-weighted Boltzmann probability $\mathbb{P}^{[x]}_S(s)$ of each sequence s, respectively such that

$$\mathcal{Z}^{[x]}_S = \sum_{|s|=n} \mathcal{B}^{[x]}_S(s) \qquad \text{and} \qquad \mathbb{P}^{[x]}_S(s) = \frac{\mathcal{B}^{[x]}_S(s)}{\mathcal{Z}^{[x]}_S}. \tag{2}$$

2.2 Linear-time stochastic sampling algorithm for the GC-weighted-Boltzmann ensemble

Let us now describe a linear-time algorithm to sample sequences at random in the GC-weighted Boltzmann distribution. This algorithm follows the general principles of the recursive approach to random generation (Wilf, 1977), pioneered in the context of RNA by the `SFold` algorithm (Ding and Lawrence, 2003). The algorithm starts by precomputing the partition function restricted to each substructure occurring in the target structure, and then performs a series of recursive stochastic backtracks, using precomputed values to decide on the probability of each alternative.

2.2.1 Precomputing the GC-weighted partition function Firstly, a dynamic programming algorithm computes $\mathcal{Z}^{[a,b]}_{N,S}$ the GC-weighted partition function (the dependency in x is omitted here for the sake of clarity) for a structure S, assuming its (previously chosen) flanking nucleotides are a and b respectively, either forming a closing base-pair ($N = \mathsf{T}$) or not ($N = \mathsf{F}$). Remark that the empty structure only supports the empty sequence, having energy 0, so one has

$$\mathcal{Z}^{[a,b]}_{\mathsf{T},\varepsilon} = \mathcal{Z}^{[a,b]}_{\mathsf{F},\varepsilon} = e^{-0/RT} = 1. \tag{3}$$

The general recursion scheme consists in three different terms, depending on the first position in S:

Case 1. First position is unpaired ($S = \bullet S'$):

$$\mathcal{Z}^{[a,b]}_{\mathsf{T},\bullet S'} = \mathcal{Z}^{[a,b]}_{\mathsf{F},\bullet S'} := \sum_{a'\in\mathcal{B}} x^{\#\mathsf{gc}(a')} \cdot \mathcal{Z}^{[a',b]}_{\mathsf{F},S'}; \tag{4}$$

Case 2. First position is paired with last position ($S = (S')$), stacking onto a pre-existing exterior pair ($N = \mathsf{T}$):

$$\mathcal{Z}^{[a,b]}_{\mathsf{T},(S')} := \sum_{a',b'\in\mathcal{B}^2} x^{\#\mathsf{gc}(a'.b')} \cdot e^{\frac{-E^{\beta}_{ab\to a'b'}}{RT}} \cdot \mathcal{Z}^{[a',b']}_{\mathsf{T},S'}; \tag{5}$$

Algorithm 1: $\mathtt{SB}_x\ (a, b, N, S)$

$r \leftarrow \mathtt{Random}\left(\mathcal{Z}_{N,S}^{[a,b]}\right)$ // Random real in $[0, \mathcal{Z}_{N,S}^{[a,b]}[$
switch do
  case $S = \varepsilon$ **return** ε; // Empty structure
  case $S = \bullet\, S'$ // First position is unpaired
    for $a' \in \mathcal{B}$ **do**
      $r \leftarrow r - x^{\#\mathsf{gc}(a')} \cdot \mathcal{Z}_{\mathsf{F},S'}^{[a',b]}$
      if $r < 0$ **then return** $a'.\mathtt{SB}_x(a', b, \mathsf{F}, S')$
  case $S = (\, S'\,)$ **and** $N = \mathsf{T}$ // Extremities are involved in stacking base pair
    for $(a', b') \in \mathcal{B} \times \mathcal{B}$ **do**
      $r \leftarrow r - x^{\#\mathsf{gc}(a'.b')} \cdot e^{-E^{\beta}_{ab \to a'b'}/RT} \cdot \mathcal{Z}_{\mathsf{T},S'}^{[a',b']}$
      if $r < 0$ **then return** $a'.\mathtt{SB}_x(a', b', \mathsf{T}, S').b'$
  otherwise // First position is paired without a stacking pair
    // $S = (\, S'\,)\, S''$
    for $(a', b') \in \mathcal{B} \times \mathcal{B}$ **do**
      $r \leftarrow r - x^{\#\mathsf{gc}(a'.b')} \cdot e^{\frac{-E^{\beta}_{\varnothing \to a'b'}}{RT}} \cdot \mathcal{Z}_{\mathsf{F},S'}^{[a',b']} \cdot \mathcal{Z}_{\mathsf{T},S''}^{[b',b]}$
      if $r < 0$ **then return** $a'.\mathtt{SB}_x\,(a', b', \mathsf{T}, S')\,.b'.\mathtt{SB}_x\,(b', b, \mathsf{F}, S'')$

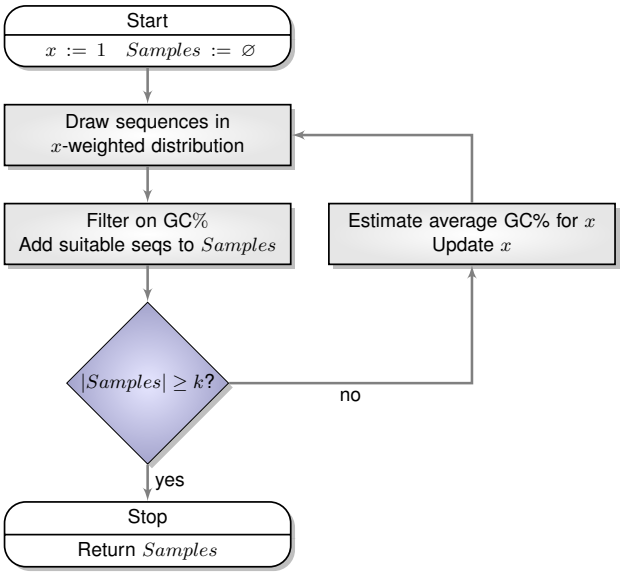


Fig. 2: General workflow of our adaptive sampling algorithm (Waldispühl and Ponty, 2011).

Case 3. First position is involved in a base-pair ($S = (\, S'\,)\, S''$), which is not stacking onto an exterior base-pair ($N = \mathsf{F}$ or $S'' \neq \varepsilon$):

$$\mathcal{Z}_{N,\,(\,S'\,)\,S''}^{[a,b]} := \sum_{a',b' \in \mathcal{B}^2} x^{\#\mathsf{gc}(a'.b')} \cdot e^{\frac{-E^{\beta}_{\varnothing \to a'b'}}{RT}} \cdot \mathcal{Z}_{\mathsf{T},S'}^{[a',b']} \cdot \mathcal{Z}_{\mathsf{F},S''}^{[b',b]}. \tag{6}$$

Remark that the number of combinations of a, b and N remains bounded by a constant, thus the complexity of computing $\mathcal{Z}_{N,S}^{[a,b]}$ mainly depends on the values taken by S upon subsequent recursive calls. Such values are entirely determined by S at any given step of the recursion, and their dependency can be summarized in a tree having $\Theta(|S|)$. Therefore, the computation of $\mathcal{Z}_{N,S^*}^{[a,b]}$ requires $\Theta(n)$ time and space using dynamic-programming.

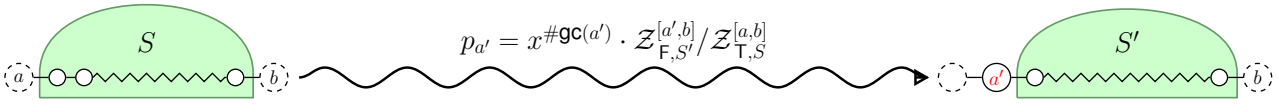
2.2.2 Stochastic backtrack Once the GC-weighted partition functions have been computed and memorized, a stochastic backtrack starts from the target structure S^* with any exterior bases $[a, b]$ and no nesting base-pair, corresponding to a call $\mathtt{SB}_x$ $(\varnothing, \varnothing, \mathsf{F}, S^*)$ to Algorithm 1. At each step, a suitable assignment for one or several positions is chosen, using probabilities derived from the precomputation, as illustrated by Figure 1. One or several recursive calls over the appropriate substructures are then performed. On each recursive call, the algorithm assigns at least one nucleotide to a – previously unassigned – position. Moreover, the number of executions of each loops is bounded by a constant. Consequently, the complexity of Algorithm 1 is in $\Theta(n)$ time and space.

2.2.3 Self-adaptive sampling strategy Let us remind that our goal is to produce a set of sequences whose GC-content matches a prescribed value gc. An absolute tolerance κ may be allowed, so that the GC-content of any valid sequence must fall in $[gc - \kappa, gc + \kappa]$. Since sequences of arbitrary GC-content may be generated by Algorithm 1, we use a rejection-based approach (Bodini and Ponty, 2010), previously adapted by the authors in a similar context (Waldispühl and Ponty, 2011). This gives an algorithm which generates k valid sequences in expected time $\Theta(k \cdot n\sqrt{n})$ when $\kappa = 0$ (or $\Theta(k \cdot n)$ when κ is a positive constant) and memory in $\Theta(k \cdot n)$. A complete analysis of the rejection process can be found in an earlier contribution (Waldispühl and Ponty, 2011), but let us briefly outline the approach, and the main arguments used to establish its complexity.

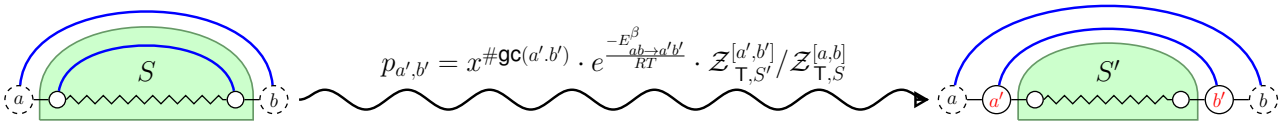
As summarized by Figure 2, our adaptive sampling approach simply generates sets of sequences by repeatedly running the stochastic backtrack algorithm. The average GC-content induced by the current value of the x parameter, can then be adequately estimated from the sample, or computed exactly using recent algorithmic advances (Ponty and Saule, 2011). The set of sequences is filtered to only retain valid sequences. The value of the parameter x is then adapted to match the average GC-content (induced by the value of x) with the targeted one. It can be shown that the expected GC-content is a continuous and strictly increasing monotonic function of x, whose limits are 0 when $x = 0$ and n when $x \to +\infty$. Consequently, for any targeted GC-content $gc \in [0\%, 100\%]$, there exists a unique value x_{gc} such that generated sequences feature, on the average, the right GC-content. In practice, a simple binary search (Waldispühl and Ponty, 2011) is used in our implementation, and typically converges after very few iterations. An optimal value for x can also be derived analytically using interpolation after $\Theta(n)$ evaluations of $\mathcal{Z}_{i,j}^{[a,b]}$ for different candidate values of x, as previously noted (Waldispühl and Ponty, 2011) and could be implemented using the Fast-Fourier Transform (Senter *et al.*, 2012).

2.2.4 Overall complexity It was previously established (Waldispühl and Ponty, 2011) that, for each value of x, there exists

Case 1: First position is unpaired.



Case 2: Extremities are paired, surrounded by another base-pair, forming a stacking base-pair.



Case 3: First position in paired to some position, but not involved in a stacking pair.

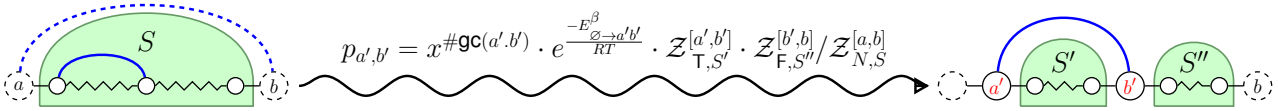


Fig. 1: Stochastic backtrack procedure for a given substructure S: Either the first position is left unpaired (top), a base-pair is formed between the two extremities, stacking onto an exterior base-pair (middle), or paired without creating a stacking, defining two regions on which subsequent recursive calls are needed (bottom). For the empty structure (omitted here), the empty sequence is returned. Positions indicated in red are assigned at the current stage of the backtrack.

constants μ_x and σ_x such that the distribution of GC-content asymptotically converges towards a normal law having expectation in $\mu_x \cdot n \cdot (1 + o(1))$ and standard deviation in $\sigma_x \cdot \sqrt{n} \cdot (1 + o(1))$. Furthermore, the distribution of GC-content is highly concentrated, as asserted by its limited standard deviation, therefore the expected number of attempts required to generate a valid sequence when $\kappa = 0$ (resp. $\kappa \in \Omega(1/\sqrt{n})$) grows like $\Theta(\sqrt{n})$ (resp. $\Theta(1)$, i.e. a constant), leading to the announced complexities. Formally, since a suitable weight x must be recomputed for each targeted structure and GC-content, then the number M of iterations required for the converge can be accounted for explicitly, leading to time complexities in $\Theta((M + \sqrt{n}) \cdot k \cdot n)$ (if $\kappa = 0$, i.e. without any tolerance) and $\Theta(M \cdot k \cdot n)$ (if $\kappa > 0$).

2.3 Postprocessing unpaired regions: A local/global (glocal) hybrid approach

Due to our simplified energy model, unpaired regions are not subject to design constraints other than the GC-content, leading to modest probabilities for refolded design candidates to match the targeted structure. To improve these performances and test the complementarity of our global sampling approach with previous contributions based on local search, we used the `RNAinverse` software to redesign unpaired regions. We specified a constraint mask to prevent stacking base-pairs from being modified and, whenever necessary, reestablished their content *a posteriori*, as `RNAinverse` has been witnessed to take some liberties with constraints masks. As shown in Table 1 (Supplementary material), this postprocessing does not drastically alter the GC-content, so the glocal approach reasonably addresses the constrained GC-content design problem.

3 RESULTS

3.1 Implementation

Our software, `IncaRNAtion`, was implemented in `Python 2.7`. We used `RNAinverse` from the *Vienna Package 2.0* (Hofacker *et al.*, 1994). All time benchmarks were run on a single AMD Opteron(tm) 6278 Processor at 2.4 GHz with cache of 512 KB. The penalty β, associated with invalid base-pairs, was set to 15.

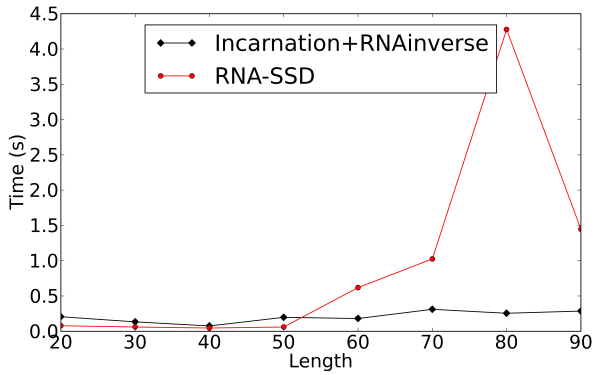


Fig. 3: Average time in seconds to generate one sequence for `IncaRNAtion` and `RNAinverse`.

Figure 3 presents the average times spent running `IncaRNAtion` +`RNAinverse` to generate one sequence with the required GC-content. As expected, the time grows linearly in function of the length of the structures for `IncaRNAtion`.

3.2 Dataset

To evaluate the quality of our method, we used secondary structures from the RNA STRAND database (Andronescu *et al.*, 2008). Those are known secondary structure from a variety of organisms. We considered a subset of 50 structures selected by Levin *et al.* (2012), whose length ranges between 20 and 100 nucleotides. To ease the visualization of results, we clustered together structures having similar length, stacks density and proportion of free nucleotides in loops, leading to distributions of structures shown in Figure 4.

3.3 Design

We ran our method as follows. First, we sampled approximately 100 sequences per structure. Then, we use these sequences as seed in `RNAinverse`. Finally, we computed the MFE with the `RNAfold` program from the *Vienna Package 2.0* (Hofacker *et al.*, 1994).

Before starting our benchmark, we asses the need for our methods and performed an analysis of the GC-content drift achieved with state-of-the-art software. Using our dataset of 50 structures, we generated 100 samples per structure with classical softwares who do not control the GC-content. Namely, `RNAinverse`, `INFO-RNA`, `NUPACK:Design` and `Frnakenstein`. We show the distribution of the GC-content of the sequences produced with these softwares in Fig. 5 those distributions.

As anticipated, we observe a clear bias toward high GC-contents and a complete absence of sequence with less than 30% of GC. This striking results motivates a need for methods that enable to explicitly control the GC-content and more precisely that enable to design sequences with low GC-content (i.e. 30% or less). In order to provide a complete overview of the performance of `IncaRNAtion`, we provide additional statistics for these software in the supplementary material.

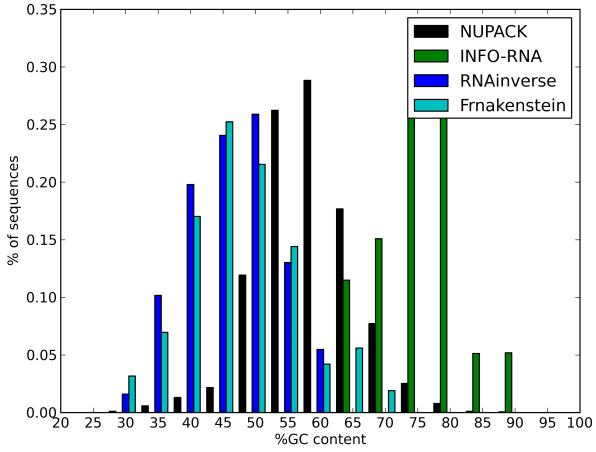


Fig. 5: Overall GC-content distribution for sequences designed using `RNAinverse`, `INFO-RNA`, `NUPACK:Design` and `Frnakenstein` folding in the desired structure.

3.4 Success rate

We started by estimating the success rate of our methodology and computed the percentage of sequences with a MFE structure identical to the target secondary structure. Figure 6 shows our results. We clearly see that before the post-processing step (i.e. `RNAinverse`) the sequences sampled by `IncaRNAtion` have a low success rate (first row). As mentioned earlier, this could be explained by the fact that no selection criterion has been at this stage applied to unpaired nucleotides. Remarkably, after the local search optimization (with `RNAinverse`) of nucleotides in unpaired regions (second row), we observe a dramatic improvement of our success rate. As expected, we observed that length is, in general, not a good predictor for the hardness of designing a structure. Instead, a high number of free nucleotides in the structure seems to be a good measure of the hardness of its design. Similarly, these data also show that designing sequences with low GC-content is challenging for all types of targets.

We investigated further the quality of the sequences generated by `IncaRNAtion`. In particular, we estimated the capacity of our methods to generate "good" sequences with desired folding capabilities regardless of the property to fold exactly into the target structure. In Figure 7, we show the ratio of well predicted base pairs in the MFE structure of our sampled sequences. As above, we can observe that, in all cases, the sequences that are the hardest to design are those with an extremely low GC-content. Indeed, the energetic contribution of the base pairs to the stability of the structure is weaker. Interestingly, we also notice that the most accurate sequences yield a GC-content of $70 \pm 10\%$. Overall, we observe that all our samples have good folding properties, and that there is a correlation between the "precision" of the samples and the hardness of the design.

We noticed a highly decreased structural sensitivity for the sequences with 15% free nucleotides in the loops. However, one must remain careful interpreting this observation, as the structures within this class all originate from the PDB, and are relatively small (for the complete STRAND DB, the average length is $\sim$ 526nts, compared to $\sim$ 38nts around 15% unpaired bases).

3.5 Properties of designed sequences

In this section, we further analyze the generated sequences with a MFE structure that folds into the target structure.

A desirable feature in sequence design, is to produce samples with a high sequence diversity and stable secondary structure. Therefore, in the following we will use two useful measures which are the sequence identity of the samples, and the Boltzmann probability of the target structure in the low energy ensemble.

The sequence identity is defined over a set $\mathcal{S}$ of aligned sequences (in our case, all sequences have the same length and can be trivially aligned) as :

$$\sum_{s^1,s^2 \in \mathcal{S}\times\mathcal{S}} \left(\frac{1}{|s^1|} \sum_{\substack{i \\ s^1_i \equiv s^2_i}} 1 \right) \qquad \text{Seq. identity} \qquad (7)$$

where s_i is the nucleotide at position i in sequence s. Intuitively, this measure captures the diversity of sequences generated by a given method. Next, the Boltzmann frequency is defined, for a structure S

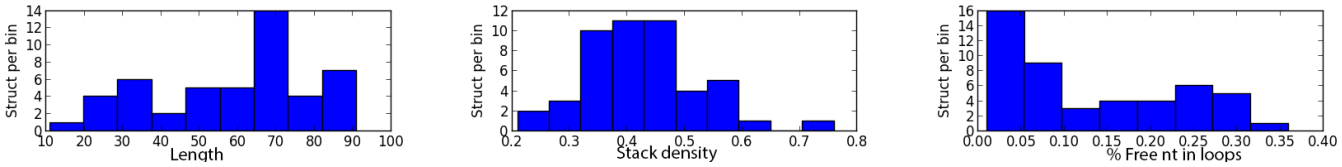


Fig. 4: Number of secondary structures per bin, according to our three clustering criteria.

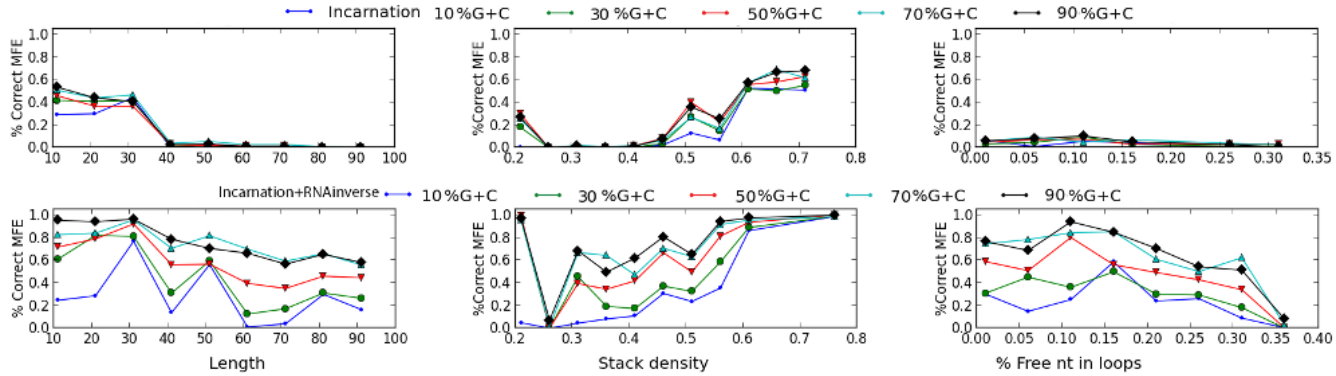


Fig. 6: Success rate `IncaRNAtion` before after after `RNAinverse` post-processing. The first row shows the percentage of sampled sequences folding into the target when using only `IncaRNAtion`. The second shows after processing previous results with `RNAinverse`.

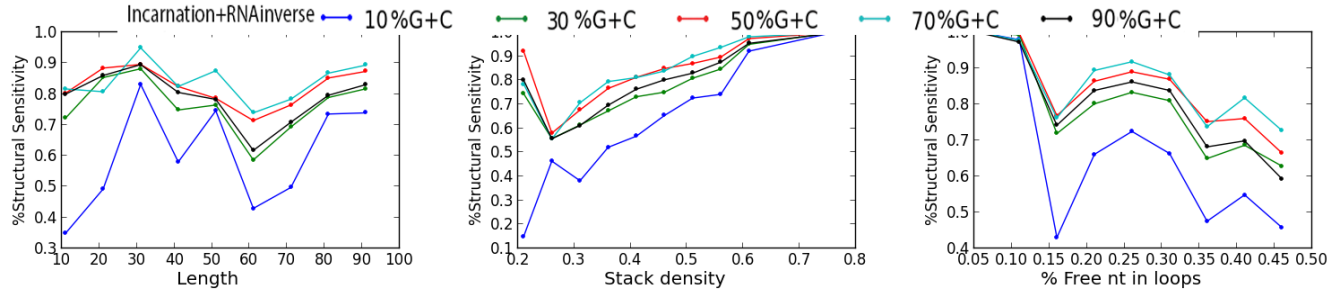


Fig. 7: Structural sensitivity (i.e. # well predicted base pairs / # base pairs in target) of the sampled sequences MFE.

and a sequence s as:

$$e^{\frac{-E(s,S)}{RT}} / \mathcal{Z}^s \qquad \text{Frequency} \qquad (8)$$

where $\mathcal{Z}^s$ is the partition function of sequence s. This measure tells us how dominant is a structure S in the Boltzmann ensemble of structures over a sequence s. A high value implies a stable structure. We compute this frequency with `RNAfold` from the `Vienna Package 2.0` (Hofacker *et al.*, 1994).

Figure 8 shows the number of solutions generated (i.e. sequences with a MFE structure identical to the target structure). Here, we note that low GC-contents have a strong (negative) influence on the number of sequences generated, and in parallel also affect negatively the sequence diversity. This observation emphasizes the difficulty to design sequences with low GC-content. Once again, large percentages of free nucleotides increase the difficulty of the task.

The thermodynamical stability of the target structure on the designed sequence is another important property when estimating the performance of RNA design algorithms. We estimate the quality of our solutions in Figure 9. First, we observe a slow decline of the structure stability (i.e. the frequency) when the target structure increases in size. Yet, for an average GC-content, the frequency stays over 10% even at size of 100 nucleotides. Next, we note that for the most difficult target structures (i.e. the longer ones or those with high percentages of unpaired nucleotides in loops) the GC-content have a limited (almost null) influence on the stability of the target structure on the designed sequence. By contrast, this is less true for easiest and small structures with only few free nucleotides in internal loops.

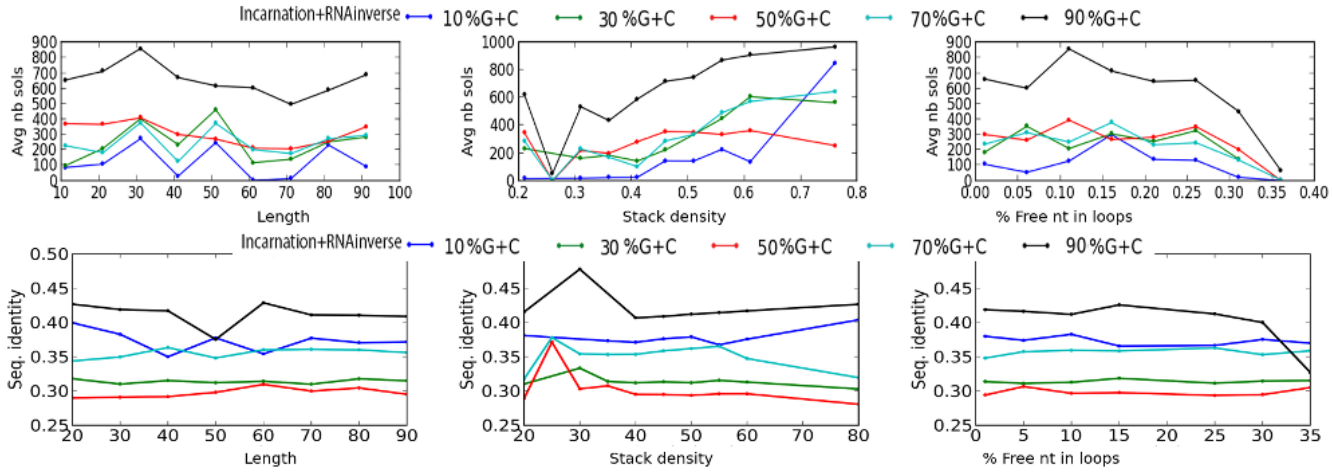


Fig. 8: Number of solutions generated with `IncaRNAtion` +`RNAinverse` on the first row and their average sequence identity on the second.

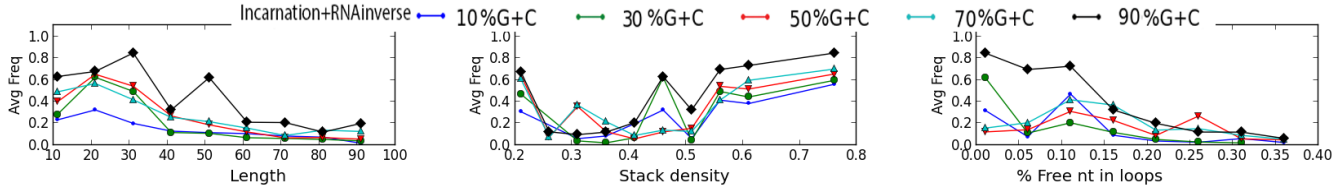


Fig. 9: Thermodynamical stability of the target structure. The curves report the average Boltzmann probability of the target structure (which is also the MFE structure) at various GC-contents w.r.t. the length of the target (left), density of stacked base pairs (centre) and number of unpaired nucleotides in loops (right).

3.6 Global sampling vs Local search vs Glocal approach

To conclude this study, we estimate the impact of the design methodology on the performances. More precisely, we aim to determine the merits of a global sampling approach (`IncaRNAtion`), compared to a glocal procedure (`IncaRNAtion` + `RNAinverse`) and a local search methodology (`RNA-SSD`). To our knowledge, `RNA-SSD`, beside `IncaRNAtion`, is the only software that implements an explicit control of the GC-content.

Here, we compare the running time and the sequence diversity of the solutions produced by each software. In addition, we focus on the design of sequences with low GC-contents (30% and less) as they are almost impossible to design with classical software (See Figure 5).

Figure 3 shows the running time of each software. These data demonstrate the efficiency and scalability of our techniques. In particular, this figure suggests that our strategy has the potential to be applied efficiently for designing sequences on long (and difficult) target secondary structures at low GC-content– A task that could have not been achieved before due time requirements.

Next, we show in Figure 10 the average sequence identity achieved by the various methods. Our results show that at extremely low GC-contents (i.e. 10%), `IncaRNAtion` slightly outperforms `RNA-SSD` while this advantage becomes less evident when the GC-content increases. Our experiments on higher GC-contents (i.e. 50% and above) showed that our glocal strategy and the local search approach perform similarly. Similarly, we did not find any clear evidence that a global, local or glocal approach outperforms others when we compare at the thermodynamical stability of the target structure (data not shown).

4 CONCLUSION

In this article, we described a novel algorithm, `IncaRNAtion`, for the RNA secondary structure design problem, i.e. the design of an RNA sequence adopting a predefined secondary structure as its minimal free-energy fold. Implementing a global sampling approach, it optimizes affinity towards the target secondary structure, while granting the user full control over the GC-content of the resulting sequences. This extended control does not necessarily induce additional computational demands, and we showed the linear complexity of both the preprocessing stage and the generation of candidate sequences for the design, allowing for the design of larger and more complex secondary structures in a matter of minutes on a single processor (e.g. $\sim$28 mins for 100 candidate sequences for a $\sim$1500nts 16s rRNA). We evaluated the method on a benchmark

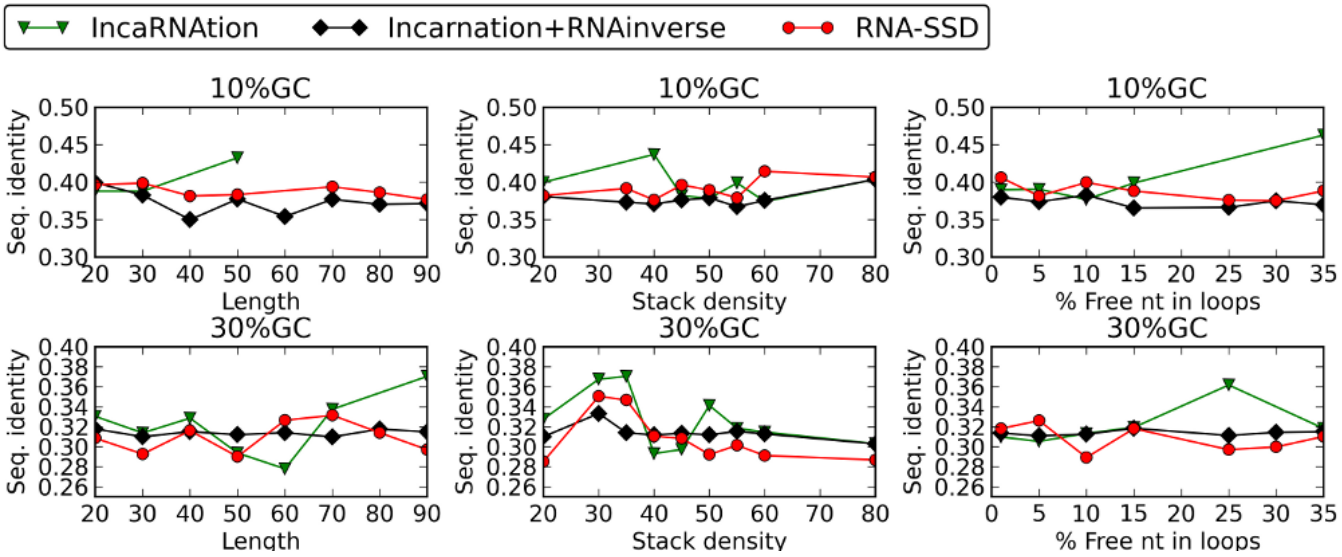


Fig. 10: Sequence identity of `IncaRNAtion` and `RNAinverse` for 10 and 30% of GC.

composed of target secondary structures extracted from the RNA STRAND database. We observed good overall success rate, with the notable exception of very low targeted GC-content (10%), and a good to excellent entropy within designed candidates. Finally, we implemented an hybrid approach, using the `RNAinverse` software as a post-processing step for unpaired regions. This approach greatly increased the success rate of the method, allowing for the design of highly diverse candidates for almost all of the structures in our benchmark, while largely preserving the targeted GC-content.

In the future, we would like to complement this study by further investigating the potential of hybrid local/global – or *glocal* – approaches. A global sampling approach would capture the positive aspects of design, optimizing affinity towards a given structure while allowing the specification of expressive systems of constraints. Designed sequences would serve as a seed for a restricted local approach which, by breaking unwanted symmetries, would perform the negative part of the design, while ideally maintaining obedience to the constraints. Another perspective of this work is the incorporation of the full Turner energy model, which should in principle yield better designs for unpaired regions.

REFERENCES

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

Andronescu, M., Bereg, V., Hoos, H., and Condon, A. (2008). RNA STRAND: the RNA secondary structure and statistical analysis database. *BMC bioinformatics*, **9**(1), 340.

Avihoo, A., Churkin, A., and Barash, D. (2011). RNAexinv: An Extended Inverse RNA Folding from Shape and Physical Attributes to Sequences. *BMC Bioinformatics*, **12**(1), 319.

Bodini, O. and Ponty, Y. (2010). Multi-dimensional Boltzmann Sampling of Languages. In *DMTCS Proceedings*, number 01 in AM, pages 49–64, Vienne, Autriche.

Busch, A. and Backofen, R. (2006). INFO-RNA–a fast approach to inverse RNA folding. *Bioinformatics*, **22**(15), 1823–31.

Chang, A. L., Wolf, J. J., and Smolke, C. D. (2012). Synthetic RNA switches as a tool for temporal and spatial control over gene expression. *Curr Opin Biotechnol*, **23**(5), 679–88.

Ding, Y. and Lawrence, E. (2003). A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Res*, **31**(24), 7280–7301.

Garcia-Martin, J. A., Clote, P., and Dotu, I. (2013). RNAiFold: A constraint programming algorithm for RNA inverse folding and molecular design. *Journal of Bioinformatics and Computational Biology*.

Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, S., Tacker, M., and Schuster, P. (1994). Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie*, **125**, 167–188.

Levin, A., Lis, M., Ponty, Y., O'Donnell, C. W., Devadas, S., Berger, B., and Waldispühl, J. (2012). A global sampling approach to designing and reengineering rna secondary structures. *Nucleic Acids Res*, **40**(20), 10041–52.

Lyngsø, R. B., Anderson, J. W., Sizikova, E., Badugu, A., Hyland, T., and Hein, J. (2012). Frnakenstein: multiple target inverse RNA folding. *BMC Bioinformatics*, **13**, 260.

Ponty, Y. and Saule, C. (2011). A Combinatorial Framework for Designing (Pseudoknotted) RNA Algorithms. In *WABI - 11th Workshop on Algorithms in Bioinformatics - 2011*, Saarbrucken, Allemagne.

Reinharz, V., Ponty, Y., and Waldispühl, J. (2013). A linear inside-outside algorithm for correcting sequencing errors in structured rna sequences. In *Proceeding of the 17th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2013)*.

Rodrigo, G., Landrain, T. E., and Jaramillo, A. (2012). De novo automated design of small RNA circuits for engineering synthetic riboregulation in living cells. *Proc Natl Acad Sci U S A*, **109**(38), 15271–6.

Senter, E., Sheikh, S., Dotu, I., Ponty, Y., and Clote, P. (2012). Using the fast fourier transform to accelerate the computational search for RNA conformational switches. *PLoS ONE*, **7**(12), e50506.

Turner, D. H. and Mathews, D. H. (2010). NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic acids research*, **38**(Database issue), D280–2.

Waldispühl, J. and Ponty, Y. (2011). An unbiased adaptive sampling algorithm for the exploration of RNA mutational landscapes under evolutionary pressure. *Journal of Computational Biology*, **18**(11), 1465–79.

Waldispühl, J., Devadas, S., Berger, B., and Clote, P. (2008). Efficient Algorithms for Probing the RNA Mutation Landscape. *PLoS Computational Biology*, **4**(8), e1000124.

Wilf, H. S. (1977). A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. *Advances in Mathematics*, **24**, 281–291.

Zadeh, J. N., Wolfe, B. R., and Pierce, N. A. (2011). Nucleic acid sequence design via efficient ensemble defect optimization. *J Comput Chem*, **32**(3), 439–52.

5 SUPPLEMENTARY DATA

5.1 Benchmark other softwares

To evaluate the performances of `IncaRNAtion`, we benchmark a set of classical softwares lacking GC-content control. Those are `RNAinverse`, `INFO-RNA`, `NUPACK:Design` and `Frnakenstein`. We present in Fig. 11 the average sequence identity and frequency for sequences generated them.

5.2 Benchmark `IncaRNAtion` + `RNAinverse`

To emphasize the usefulness of processing `IncaRNAtion` sequences with `RNAinverse`, we present the number of structures for which at least one sequence was generated with the desired MFE in Figure. 12

5.3 Limited impact on GC of local-search postprocessing of `IncaRNAtion` output

Since local search approaches tend to experience a bias towards GC-rich regions, it could be expected that our glocal approach, by postprocessing unpaired regions using a local search algorithm, would suffer from such a drift. However, as summarized in Table 1, we observed that the local search heuristic used to design nucleotides in loop regions has a very limited impact on the GC-content. For each class of GC-content, we reported the observed GC-content in the sequence initially generated by `IncaRNAtion`, and the observed GC-content after the `RNAinverse` postprocessing (as defined in Section 2.3). Our results show that the GC-content is relatively well conserved (less than 6% variation), with a general tendency of the postprocessing step to bring the GC-content back to 50%.

Target GC-content (%)	GC-content (%) of designed sequences	
	`IncaRNAtion` (Global)	`IncaRNAtion` + `RNAinverse` (Glocal)
10%	15%	21% ↗ 6%
30%	30%	33% ↗ 3%
50%	48%	49% ↗ 1%
70%	71%	69% ↘ 2%
90%	83%	78% ↘ 5%

Table 1. Observed GC-content of solutions returned by `IncaRNAtion` (2nd column) and after the application of the local search postprocessing (3rd column).

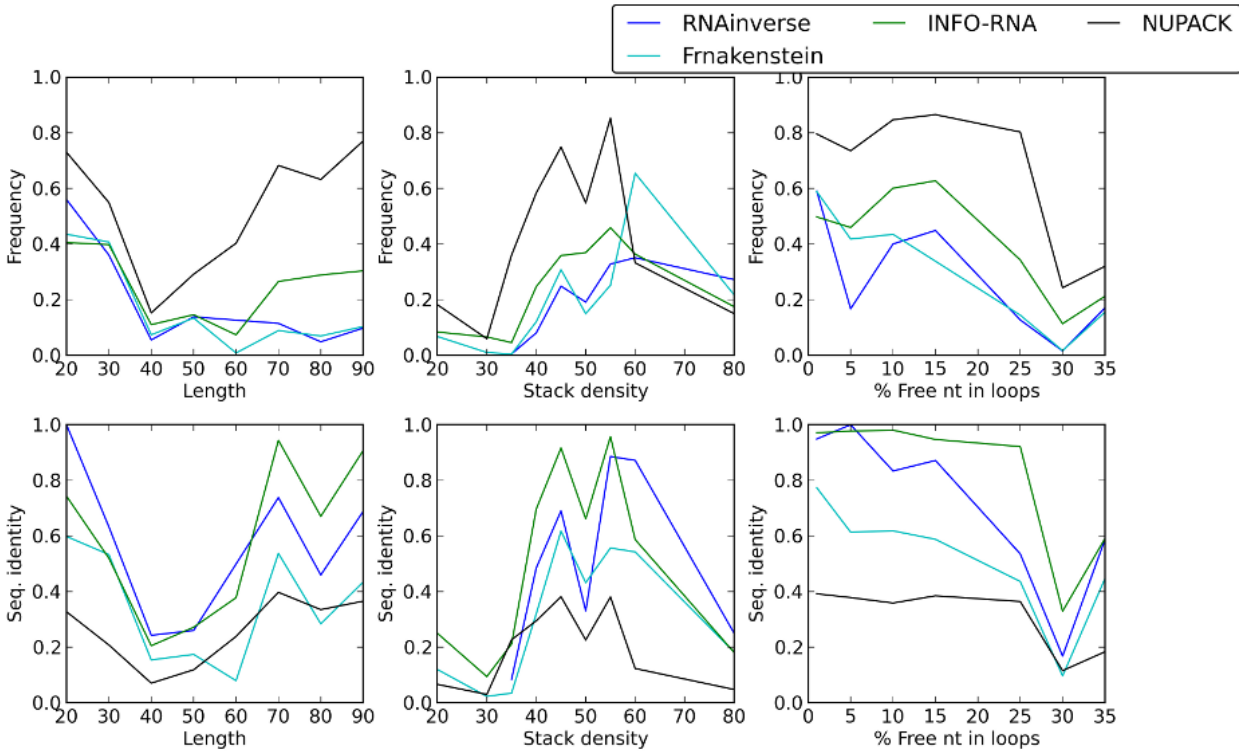


Fig. 11: The average sequence identity and frequency for softwares without GC-content control.

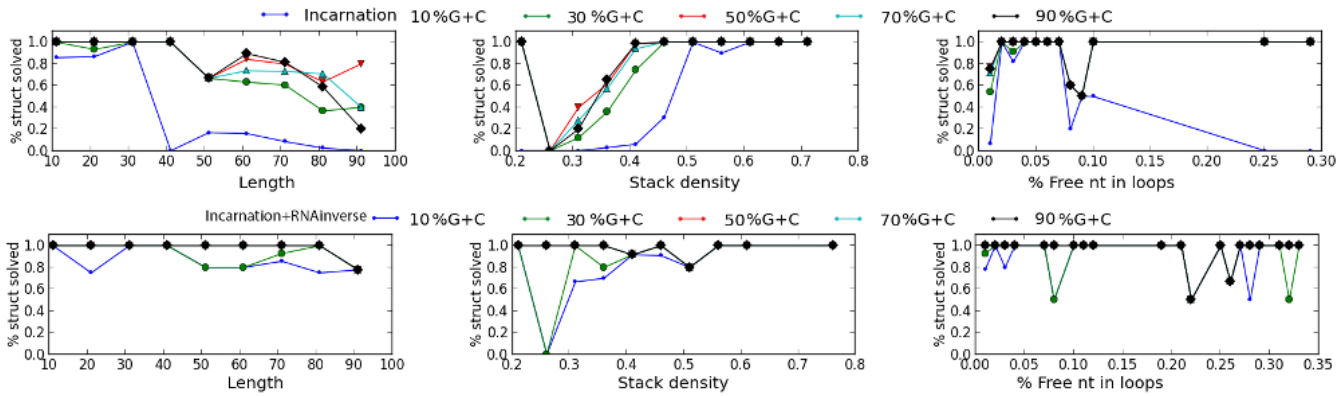


Fig. 12: The first row shows the number of structures for which one generated sequence has the structure as MFE when only using `IncaRNAtion`. The second row shows when we process `IncaRNAtion` results with `RNAinverse`.