Supplementary Material

antaRNA - Ant Colony Based RNA Sequence Design

Kleinkauf Robert [1], Mann Martin [1], Backofen Rolf [1,2,3,4,*]

[1]Bioinformatics Group, Department of Computer Science, University of Freiburg, Georges-Köhler-Allee 106, 79110 Freiburg, Germany
[2]Center for Biological Signaling Studies (BIOSS), University of Freiburg, Germany
[3]Center for Biological Systems Analysis (ZBSA), University of Freiburg, Germany
[4]Center for non-coding RNA in Technology and Health, University of Copenhagen, Grønnegårdsvej 3, DK-1870 Frederiksberg C, Denmark

Contents

*To whom correspondence should be addressed.Tel: +49 (0) 761 / 203 - 7461 ; Fax: +49 (0) 761 / 203 - 7462; Email: backofen@informatik.uni-freiburg.de

S1 Formulae

S1.1 Structure Constraint Definition

The structure constraint $\mathbb{C}^{str}$ is defined by the three-tuple $(\mathbb{C}^{str}_P, \mathbb{C}^{str}_U, \mathbb{C}^{str}_B)$ with:

$$\begin{aligned}
\mathbb{C}^{str}_P &= \{\, (i,j) \mid 1 \leq i < j \leq n \} && (1)\\
&\text{explicit set of nested base pairs that have to be formed}\\
\mathbb{C}^{str}_U &= \{\, i \mid 1 \leq i \leq n \} && (2)\\
&\text{explicit set of indices that have to be unpaired}\\
\mathbb{C}^{str}_B &= \{\mathbb{C}^{str}_{B_1}, \ldots, \mathbb{C}^{str}_{B_k}\} && (3)\\
&\text{set of block index sets } \mathbb{C}^{str}_{B_l} \text{ for implicit structure constraints with}\\
&\mathbb{C}^{str}_{B_l} = \{\, i \mid 1 \leq i \leq n \}
\end{aligned}$$

The according index sets have to constrain the whole sequence, i.e.

$$\begin{aligned}
\forall_{1 \leq i \leq n} : \quad & \exists_{(k,l) \in \mathbb{C}^{str}_P} : k = i \vee l = i\\
\vee \quad & i \in \mathbb{C}^{str}_U\\
\vee \quad & \exists_{\mathbb{C}^{str}_{B_l} \in \mathbb{C}^{str}_B} : i \in \mathbb{C}^{str}_{B_l}
\end{aligned} \tag{4}$$

while each index is to be constrained only once:

$$\begin{aligned}
& \forall_{\mathbb{C}^{str}_{B_l} \neq \mathbb{C}^{str}_{B_r} \in \mathbb{C}^{str}_B} : \mathbb{C}^{str}_{B_l} \cap \mathbb{C}^{str}_{B_r} = \emptyset\\
\wedge \quad & \mathbb{C}^{str}_U \cap \bigcup_{\mathbb{C}^{str}_{B_l} \in \mathbb{C}^{str}_B} \mathbb{C}^{str}_{B_l} = \emptyset\\
\wedge \quad & \forall_{(i,j) \in \mathbb{C}^{str}_P} : i \notin X \wedge j \notin X\\
& \text{with } X = \mathbb{C}^{str}_U \cup \bigcup_{\mathbb{C}^{str}_{B_l} \in \mathbb{C}^{str}_B} \mathbb{C}^{str}_{B_l}
\end{aligned} \tag{5}$$

As an example consider the following constraint encoding:

```
index       12345678901234567890123
constraint  AAA(((((BB...BBB)))))AA
```

which results in the following structure constraint:

$$\begin{aligned}
\mathbb{C}^{str}_P &= \{(4,21),(5,20),(6,19),(7,18),(8,17)\}\\
\mathbb{C}^{str}_U &= \{11,12,13\}\\
\mathbb{C}^{str}_B &= \{\, \mathbb{C}^{str}_{B_1} = \{1,2,3,22,23\}, \mathbb{C}^{str}_{B_2} = \{19,20,24,25,26\} \,\}
\end{aligned}$$

Lonely base pairs in $\mathbb{C}^{str}_P$: To evaluate whether or not a solution sequence $\mathcal{S}^{sol}$ forms the target structure defined by $\mathbb{C}^{str}$, a $\mathcal{S}^{sol}$-representing solution structure P^{sol} has to be computed. This is done via *RNAfold* from the *ViennaRNA*-package in it's version 2.1.3 [1, 2], which computes the minimal free energy (mfe) structure for a given sequence according to a nearest neighbor energy model, see [3].

However, during folding, requested lonely base pairs might not occur within mfe structures, since they could be energetically unfavorable to structures. To counteract this fact, we introduce a special term within the structural distance, which deals with lonely base pairs in order to review their explicit presence within the structure. Therefore, formally, a base pair (i, j) is called a lonely base pair in structure P if $\nexists (k, l) \in P : |i - k| + |j - l| = 2$. A similar effect holds for *lonely 2-stacks*, i.e. two stacking base pairs that are not part of a consecutive helix/stacking.

Given both cases, we define the set of lonely base pairs LP for a nested structure P by

$$\begin{aligned} LP(P) = \{\, (i,j) \mid \quad & \nexists_{(k,l)\in P} : |i-k| + |j-l| = 2 \\ & \vee ((i+1, j-1) \in P \wedge (i-1, j+1) \notin P \wedge (i+2, j-2) \notin P) \\ & \vee ((i-1, j+1) \in P \wedge (i+1, j-1) \notin P \wedge (i-2, j+2) \notin P) \\ \} & \end{aligned} \tag{6}$$

S1.2 Terrain Initialization

In the following, we will detail the initial pheromonic and static heuristic values.

Sequence constraint dependent pheromone initialization: A sequence constraint $\mathbb{C}^{seq}$ defines for each sequence position j the set of allowed nucleotides. An encoding of $\mathbb{C}^{seq}_j = N$ allows for all nucleotides from Σ while a nucleotide specific setup ($\mathbb{C}^{seq}_j \in \Sigma$) demands for this nucleotide to be emitted at position j.

Since the terrain graph encodes for all possible nucleotide emissions for each position, we have to adjust the edge probabilities for edge $e_{(i\sigma, j\sigma')}$ leading to position j emitting nodes in order to avoid forbidden emissions. To this end, we will set the initial pheromone information for edges leading to forbidden emissions to zero. Note, beside the explicit sequence constraint $\mathbb{C}^{seq}$ we also face implicit sequence constraints from the combination of $\mathbb{C}^{seq}$ with the enforced base pairs from $\mathbb{C}^{str}_P$. That is, if position k is constrained to a nucleotide ($\mathbb{C}^{seq}_k \in \Sigma$) and there exists a required base pair $(k, j) \in \mathbb{C}^{str}_P$ then position j is implicitly constrained to a nucleotide complementary to $\mathbb{C}^{seq}_k$. The latter is defined by $\text{COMPL} : (\Sigma \cup \{N\}) \to \mathcal{P}(\Sigma \cup \{N\})$ given by

$$\text{COMPL}(\sigma) = \begin{cases} \{U\} & \text{if } \sigma = A \\ \{G\} & \text{if } \sigma = C \\ \{C, U\} & \text{if } \sigma = G \\ \{A, G\} & \text{if } \sigma = U \\ \Sigma & \text{if } \sigma = N \end{cases} \tag{7}$$

As an example consider a base pair $(i, j) \in \mathbb{C}^{str}_P$ and a sequence constraint $\mathbb{C}^{seq}_i = G$. This implies a sequence constraint $\mathbb{C}^{seq}_j = \text{COMPL}(G) = \{C, U\}$. Given this, the initial pheromone value for each edge is defined by

$$\forall e_{(i\sigma,j\sigma')} : \tau(e_{(i\sigma,j\sigma')}) = \begin{cases} 0 & \text{if } (\mathbb{C}_j^{seq} \neq N \wedge \sigma' \neq \mathbb{C}_j^{seq}) \text{ or} \\ & (\mathbb{C}_j^{seq} = N \wedge \exists (k,j) \in \mathbb{C}_P^{str} : \sigma' \notin \text{COMPL}(\mathbb{C}_k^{seq})) \\ 1 & \text{else} \end{cases} \tag{8}$$

GC-dependent heuristic initialization: The target vertex $v_{j\sigma'}$ of an edge $e_{(i\sigma,j\sigma')}$ defines the emitted nucleotide σ' at position j of the solution sequence $\mathcal{S}^{sol}$. The heuristic information $\eta(e_{(i\sigma,j\sigma')})$ in concert with the edge's pheromone value $\tau(e_{(i\sigma,j\sigma')})$ are incorporated into the edge probability (see sequence assembly). The part η is, in contrast to τ, a static property of an edge. Therefore, it can be used to statically adjust edge probabilities according to the targeted GC content of the solution defined by $\mathbb{C}^{\text{GC}}$ and encodes the *heuristic edge information* (see main document).

To this end, we apply a nucleotide-specific value incorporating the target GC-value $\mathbb{C}^{\text{GC}}$ as follows:

$$\forall e_{(i\sigma,j\sigma')} : \eta(e_{(i\sigma,j\sigma')}) = \begin{cases} (0.5 + \mathbb{C}^{\text{GC}}) & \text{if } \sigma' \in \{A, U\} \\ (1.5 - \mathbb{C}^{\text{GC}}) & \text{if } \sigma' \in \{G, C\} \end{cases} \tag{9}$$

For $\mathbb{C}^{\text{GC}} = 0.5$, a uniform value of 1 is defined. Deviating values result in an increase of the beneficial edges, i.e. edges that lead to vertices emmitting more or less "GC" into the solution sequence.

S1.3 Measurements for determining a sequence's quality

Determination of the final target structure: The possibility to define fuzzy implicit structure constraints results in an undefined final target structure $P^{\mathbb{C}}$ to which the representative structure P^{sol} of the designed sequence solution $\mathcal{S}^{sol}$ is compared to. The target structure $P^{\mathbb{C}}$ has to be derived from both $\mathbb{C}^{str}$ as well as the representative structure P^{sol} by the fusion of

- all enforced base pairs from $\mathbb{C}_P^{str}$ (excluding lonely base pairs),
- all base pairs from P^{sol} within implicit structure constraint blocks $\mathbb{C}_{B_l}^{str}$,
- all lonely base pairs from $\mathbb{C}^{str}$ that are present in P^{sol},
- all sequence constraint induced base pairs exclusive to P^{sol}.

That is, given the subset of lonely base pairs of a structure P by $LP(P)$ (Eq. 6), the target structure is defined by

$$\begin{aligned} P^{\mathbb{C}} &= \mathbb{C}_P^{str} \setminus LP(\mathbb{C}_P^{str}) \\ &\cup \bigcup_{\mathbb{C}_{B_l}^{str} \in \mathbb{C}_B^{str}} \left\{ (i,j) \mid i,j \in \mathbb{C}_{B_l}^{str} \wedge (i,j) \in P^{sol} \right\} \\ &\cup LP(\mathbb{C}_P^{str}) \cap P^{sol} \\ &\cup \left\{ (i,j) \mid (i,j) \in (P^{sol} \setminus \mathbb{C}_P^{str}) \wedge \mathbb{C}_i^{seq} \neq N \wedge \mathbb{C}_j^{seq} \neq N \right\}. \end{aligned} \tag{10}$$

Structural distance: The used structural distance between two RNA secondary structures P and P' represents the normalized symmetric base pairs difference of both structures, including lonely base pair penalties.

In practice, in order to compute the structural distance, we calculate the symmetric difference of the base pair sets of the target structure $P^{\mathbb{C}}$ and the structure P^{sol} of the designed sequence $\mathcal{S}^{sol}$. Since $P^{\mathbb{C}}$ does not necessarily cover all lonely base pairs from $LP(\mathbb{C}_P^{str})$ (see Eq. 10), we have to account for the compatibility of the remaining lonely base pairs within the sequence to get a complete structural distance score. That is, if a requested lonely base pair can not be formed by $\mathcal{S}^{sol}$, a penalty is added to the basic distance value. The set of incompatible lonely base pairs $P_{\text{inc}}^{\mathbb{C}} \subseteq LP(\mathbb{C}_P^{str})$ is given by

$$P_{\text{inc}}^{\mathbb{C}} = \{\ (i,j) \mid (i,j) \in LP(\mathbb{C}_P^{str}) \wedge \mathcal{S}_i^{sol} \notin \text{COMPL}(\mathcal{S}_j^{sol})\} \tag{11}$$

In the end, the distance is normalized with the length of the sequence. This defines the structural distance measure d_{str} as follows:

$$d_{str}(P^{sol}, P^{\mathbb{C}}) = \frac{|(P^{sol} \setminus P^{\mathbb{C}}) \cup (P^{\mathbb{C}} \setminus P^{sol})| + \frac{|P_{\text{inc}}^{\mathbb{C}}|}{2}}{n} \tag{12}$$

Incompatible lonely base pairs are only considered with a weight of $\frac{1}{2}$ in the distance function (Eq. 12). This is done to respect that lonely base pairs that can be formed but are missing in the solution structure are not penalized while "normal" base pairs that are missing get a full penalty. Thus, the impossible base pairs from $P_{\text{inc}}^{\mathbb{C}}$ are only penalized half.

GC-aberration: The GC-aberration d_{GC} reflects the deviation of the actual GC-content of a solution sequence $\mathcal{S}^{sol}$ from the target GC-value $\mathbb{C}^{\text{GC}}$. The actual GC-content *actGC* is the fraction of G or C nucleotides within the sequence, i.e.

$$actGC = \frac{|\ \{\ i \mid \mathcal{S}_i^{sol} \in \{G, C\}\}\ |}{n} \tag{13}$$

where n denotes the sequence's length.

Thus, possible values of *actGC* are discretized based on n such that target GC values $\mathbb{C}^{\text{GC}}$ are often not accessible. For instance, given a $\mathbb{C}^{\text{GC}} = 0.55$ and a sequence length of $n = 5$, the best *actGC* accessible are $\frac{2}{5} = 0.4$ and $\frac{3}{5} = 0.6$ leaving an intrinsic deviation of 0.15 and 0.05, respectively. The intrinsic deviation off the GC values decreases with increasing sequence length n. This is due to each nucleotides percentage of contribution to the GC content. Fig. S1 illustrates the GC shift phenomenon within one sequence example and different targeted GC values.

For this reason, we apply a correction to the target GC-content in order to measure the solutions deviation to the best accessible GC-content. This way, a GC-aberration value of $d_{\text{GC}} = 0$ is possible for all sequence lengths. Since the intrinsic deviation is not symmetric to positive or negative aberration, we use dedicated values δ_{GC}^{+} and δ_{GC}^{-}, respectively. Given the example from above, we would increase the target GC-content from $\mathbb{C}^{\text{GC}} = 0.55$ to the range $[0.4, 0.6]$ using intrinsic deviations of $\delta_{GC}^{+} = 0.05$ and $\delta_{GC}^{-} = 0.15$.

$$d_{\mathrm{GC}} = \begin{cases} \Delta_{GC} - \delta^{+}_{GC} & \text{if } \Delta_{GC} \geq 0 \\ \Delta_{GC} + \delta^{-}_{GC} & \text{if } \Delta_{GC} < 0 \end{cases} \tag{14}$$

$$\Delta_{GC} = actGC - \mathbb{C}^{\mathrm{GC}}$$
absolute deviation (15)

$$\delta^{+}_{GC} = \frac{|(\mathbb{C}^{\mathrm{GC}} * n) - \lceil \mathbb{C}^{\mathrm{GC}} * n \rceil|}{n}$$
positive intrinsic deviation (16)

$$\delta^{-}_{GC} = \frac{|(\mathbb{C}^{\mathrm{GC}} * n) - \lfloor \mathbb{C}^{\mathrm{GC}} * n \rfloor|}{n}$$
negative intrinsic deviation (17)

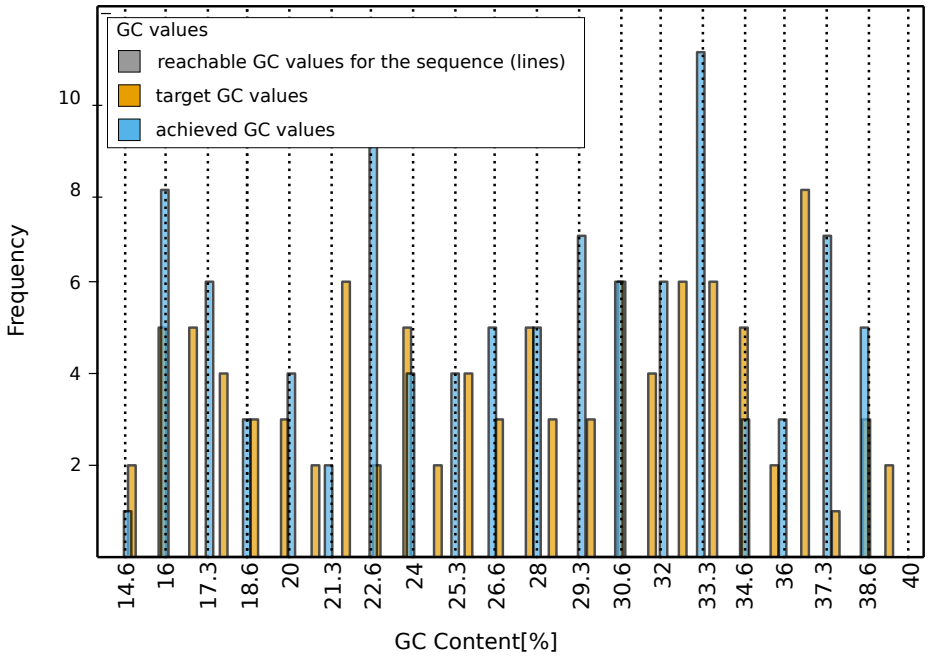

Figure S1: **GC-shift towards possible GC-content values** A sequence's GC-content can only adopt discretized values, since each nucleotide has a length-dependent contribution to the GC-value. Thus, only a limited number of GC values can be reached by design programs. For *antaRNA*, the GC-content of sequences sampled for a targeted uniform GC-content distribution shifts towards those reachable GC values. The calculation of the GC aberration (d_{GC}) is conducted with this information. Here the behaviour of *antaRNA* is demonstrated on the example structure from Fig. S2. The 100 generated GC values (yellow) are sampled from a uniform distribution in the target GC value range of $15\% - 40\%$. The actual GC values (blue) are shifted towards the possible GC values (grey lines).

Sequential distance: The sequence distance d_{seq} is calculated between the produced sequence $\mathcal{S}^{sol}$ and the defined sequence constraint $\mathbb{C}^{seq}$ if provided. It represents the ratio of constrained sequence positions that show the wrong nucleotide assignment. Unconstrained positions are ignored.

$$d_{seq} = \frac{|\{\ i \mid \mathbb{C}_i^{seq} \neq N \wedge \mathcal{S}_i^{sol} \neq \mathbb{C}_i^{seq}\ \}|}{n} \tag{18}$$

Overall quality evaluation: In order to assess the overall quality Q of the current solution sequence $\mathcal{S}^{sol}$, the three different deviation values d_{str}, d_{GC} and d_{seq} are combined. Therein, each deviation is weighted by an according factor γ in order to allow for a fine tuning of the quality estimate. The higher Q the better the solution sequence respects the constraints $\mathbb{C}^{str}, \mathbb{C}^{\mathrm{GC}}$ and $\mathbb{C}^{seq}$.

$$Q = \gamma_{str} * q(d_{str}) + \gamma_{GC} * q(d_{\mathrm{GC}}) + \gamma_{seq} * q(d_{seq}) \tag{19}$$

$$q(d) = \begin{cases} \frac{1}{d} & \text{if } d > 0 \\ \Omega & \text{else } d = 0 \end{cases} \tag{20}$$

where Ω represents a (high) default quality measure if no deviation for a measure ($d = 0$) was achieved. Within the implementation, we used the $\frac{1}{x}$ function to calculate the score of a quality. Since this function would grow without bound, a function was augmented, based on the series of $\frac{1}{x}$, which would be artificially able to intersect the y-axis, such this intersection would be $\Omega = \sqrt{5} = 2.23$.

Entropy The entropy is used as measure of diversty in this study. Given a set of equally long sequences, the entropy is calculated position wise. The entropy H for one sequence position in the sequences is calculated as

$$H = -log_2 \sum_{i=1}^{m} p_i log_2(p_i) \tag{21}$$

where m is the sequences' alphabet size and p_i denotes the frequency of the according alphabet letter in all sequences at the position of interest. In this study, we use the mean value over all sequence unconstrained positions and within all designed sequence sets.

For dinucleotide entropies, according dinucleotide frequencies have been used.

S1.4 Pheromone update

The pheromone update is comprised of two parts: First, the pheromone information of all transitions is reduced according to the global evaporation rate ρ. Afterwards, all pheromone values for transitions partaking in the solution production are increased according to the solution's quality.

The global terrain evaporation is based on the evaporation factor ρ and is given by:

$$\forall_{e \in \mathcal{E}} : \tau(e) = (1 - \rho)\tau^{\mathrm{old}}(e) \tag{22}$$

The solution $\mathcal{S}^{sol}$ dependent pheromone update increases the pheromonic information for all edges $\mathcal{E}^{sol} \subseteq \mathcal{E}$ part of the solution generation.

$$\mathcal{E}^{sol} = \{\ e_{(\bullet,1\mathcal{S}_1^{sol})}\ \} \cup \bigcup_{1<i\leq n} \{\ e_{((i-1)\mathcal{S}_{i-1}^{sol},i\mathcal{S}_i^{sol})}\ \} \tag{23}$$

The pheromone for an edge $e_{(i\sigma,j\sigma)}$ from $\mathcal{E}^{sol}$ is only increased, if the structural situation $m(j)$ of the solution structure P^{sol} at position j (given by the target vertex $v_{j\sigma}$) is in accordance with the target structure $P^{\mathbb{C}}$ at position j, i.e. if nucleotide j is in a specific base pair or single stranded in P^{sol} and $P^{\mathbb{C}}$ at the same time. The increase is given by the quality Q (Eq. 19) of the solution sequence $\mathcal{S}^{sol}$. This gives the following pheromonic update:

$$\forall e_{(i\sigma,j\sigma)} \in \mathcal{E}^{sol} : \tau(e_{(i\sigma,j\sigma)}) = \tau^{\text{old}}(e_{(i\sigma,j\sigma)}) + m(j)Q \tag{24}$$

$$m(j) \quad = \quad \begin{cases} 1 & \text{if } \exists(i,j) \in (P^{sol} \cap P^{\mathbb{C}}) \vee \exists(j,k) \in (P^{sol} \cap P^{\mathbb{C}}) \\ & \text{i.e. } j \text{ is part of a correct base pair} \\ 1 & \text{if } \nexists(i,j) \in (P^{sol} \cup P^{\mathbb{C}}) \wedge \nexists(j,k) \in (P^{sol} \cup P^{\mathbb{C}}) \\ & \text{i.e. } j \text{ is correctly single stranded} \\ 0 & \text{else} \end{cases} \tag{25}$$

The better a solution sequence respects the targeted constraints, the higher the pheromone increase.

S2 Parameter Benchmark and Selection

In order to identify the best parameter setting for *antaRNA*, we performed a grid search using the following parameter values. The selected default values are presented in the last column.

α	$\in \{0.2, 0.5, 1.0, 2.0, 4.0\}$	$\mathbf{= 1.0}$
β	$\in \{0.2, 0.5, 1.0, 2.0, 4.0\}$	$\mathbf{= 1.0}$
ρ	$\in \{0.05, 0.1, 0.2\}$	$\mathbf{= 0.2}$
γ_{str}	$\in \{0.5, 1.0, 5.0\}$	$\mathbf{= 0.5}$
γ_{GC}	$\in \{0.5, 1.0, 5.0\}$	$\mathbf{= 5.0}$
γ_{seq}	$\in \{0.5, 1.0, 5.0\}$	$\mathbf{= 1.0}$
reset potential(TP)	$\in \{25, 40, 50\}$	$\mathbf{= 50}$
convergence potential(CP)	$\in \{50, 100, 130\}$	$\mathbf{= 130}$

The grid search parameter optimization was done using the 20 structural constraints $\mathbb{C}^{str}$ from the dataset described in the main manuscript. Three different target GC-values $\mathbb{C}^{\mathrm{GC}} \in \{0.25, 0.5, 0.75\}$ were used with and without sequence constraint $\mathbb{C}^{seq}$ applied.

From this screening, the following table shows the top-ranked set of parameters. The set with lowest score was taken as standard parameter set for *antaRNA* in the study.

rank	score	α	β	ρ	γ_{str}	γ_{GC}	γ_{seq}	TP	CP
0	**0.219**	**1.0**	**1.0**	**0.2**	**0.5**	**5.0**	**1.0**	**50**	**130**
1	0.226	1.0	1.0	0.2	0.5	5.0	1.0	40	130
2	0.229	1.0	1.0	0.2	1.0	5.0	1.0	50	130
3	0.232	1.0	1.0	0.2	1.0	5.0	1.0	40	130
4	0.235	1.0	1.0	0.2	0.5	5.0	1.0	50	100
5	0.239	1.0	1.0	0.2	1.0	1.0	1.0	50	130
6	0.240	1.0	1.0	0.2	1.0	5.0	1.0	50	130
7	0.241	1.0	1.0	0.2	0.5	5.0	1.0	25	130
8	0.246	1.0	1.0	0.2	0.5	5.0	1.0	40	100
9	0.249	1.0	1.0	0.2	1.0	5.0	1.0	40	130
10	0.250	1.0	1.0	0.2	1.0	1.0	1.0	40	130

S3 Exemplary tRNA-like test structure

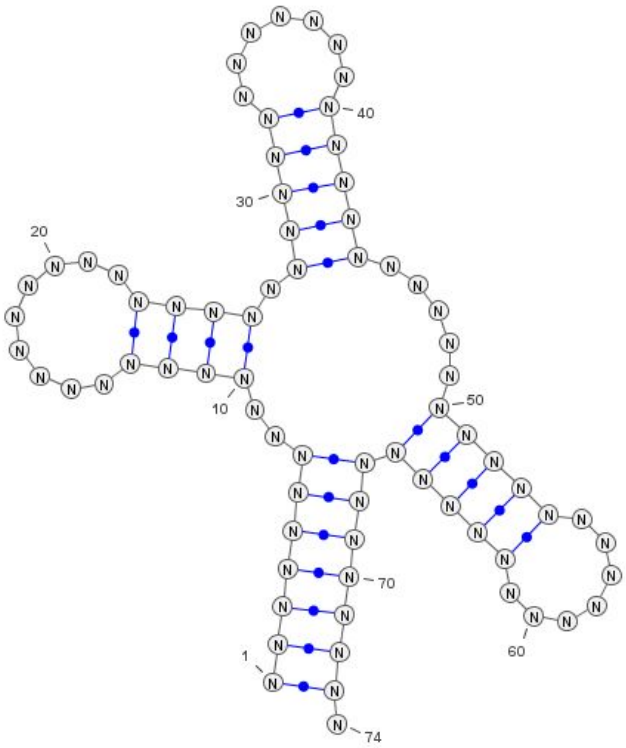

Figure S2: The tRNA-like multi-loop structure used for the visualization of the GC-bias for different RNA design programs. The according structure encoding in dot-bracket notation is `(((((((..((((.........)))).(((((.......)))))).....(((((.......)))))))))))..` The structure was drawn using the VARNA web applet v3.9 [4].

S4 Rfam dataset

The extract of the Rfam is separated into two datasets: training and test. The training set contains 20 entries, the test set 63.

Both sets (and also the whole set) are provided on the tools homepage:

http://www.bioinf.uni-freiburg.de/Software/antaRNA

Each derived constraint set of the Rfam dataset is encoded in a FASTA-like format along with additional information about different values of retreaval and intrinsic sequence/structure stats. The extra information is stored within the header. An example is given below.

```
>RF02278|AtGC=0.52|MR=0.97,CR=0.3,SRC=0.3,MAS=20,AS=54,CL=63,SR=0.25,CGCC=0.11
.......(((((((.......)))))))...(((((((.......)))))))......
NGTCTCCTCNANNNNNCNNNNNNNNNTNGATTNNNNNNNNNNNNNNNNNNNNNNNNNTNNN
```

The FASTA header encodes after the Rfam family identifier RF... the following information:

AtGC	the Rfam familiy specific alignment GC value, which is derived from the seed alignment of that family.
MR	Majority Ratio: of one explicit nucleotide over all others within a considered alignment column
CR	Constraint Rate: the fraction of $\mathbb{C}^{seq}$-constrained sequence positions
SRC	Structure Ratio Constraint: lower boundary of the ratio of base pair forming nucleotides versus single stranded nucleotides within the structure
MAS	Minimum Alignment Size
AS	Rfam family's de facto alignment size
CL	Constraint Length: the length of the derived target structure
SR	actual ratio of base pair forming nucleotides versus single stranded nucleotides within the structure
CGCC	the Constraint GC content within the actually used constraint, i.e. within the explicitly requested nucleotides

The used Rfam families are

RF00004	RF00005	RF00007	RF00031	RF00037
RF00047	RF00053	RF00056	RF00090	RF00103
RF00103	RF00167	RF00199	RF00215	RF00231
RF00237	RF00238	RF00263	RF00263	RF00398
RF00400	RF00403	RF00404	RF00406	RF00413
RF00418	RF00422	RF00424	RF00425	RF00438
RF00446	RF00451	RF00480	RF00514	RF00545
RF00553	RF00561	RF00561	RF00565	RF00568
RF00582	RF00617	RF00617	RF00639	RF00641
RF00651	RF00654	RF00657	RF00667	RF00668
RF00670	RF00670	RF00679	RF00693	RF00694
RF00706	RF00708	RF00736	RF00906	RF00951
RF01045	RF01045	RF01225	RF01234	RF01241
RF01256	RF01418	RF01690	RF01692	RF01694
RF01705	RF01709	RF01730	RF01751	RF01782
RF01797	RF01844	RF01859	RF01873	RF02002
RF02030	RF02108	RF02278		

S5 Parameters and Setting of the programs *RNAiFold* and *IncaRNAtion*

S5.1 *IncaRNAtion*

We computed the performances of *IncaRNAtion* on our local computer cluster. The tool is wrapped within a python intercase script, such that it can be subject to a submission to the cluster system. We tried to run the tool in it's standard parameters, wich was not possible when selecting the option of GC constraint. The calls included the following options and flags: The first was taken for sequence constraint foldings, the second for the usage of sequence free constraints.

```
IncaRNAtion -d [file_name]
    -a 1 -no_profile
    -s_gc [tgc] 100
    -c [Constraint Sequence]
IncaRNAtion -d [file_name]
    -a 1
    -no_profile
    -s_gc [tgc] 100
```

Since *IncaRNAtion* produces seed sequences to *RNAinverse*, the same is applied afterwards.

S5.2 *RNAiFold*

Since during the production of this article, the tool *RNAiFold* was only available through a provided webservice, the calculations of *RNAiFold* have been kindly performed by Ivan Dotu, member of the team around *RNAiFold*.

The constraints given to *RNAiFold* are:

- The allowed GC target value interval for the program was an interval [x-k%, x+k%] where x was chosen from the values [25,50,75] and k was set to 2%.
- A time out of one hour was applied.
- The internal option 'LNS' was used.
- Per execution one sequence, if any, was returned.
- The underlaying ViennaRNA Tools are in version 1.8.5. During the usage the '-d2' optin was used.

S6 Strategy Comparison: Sample and Filter vs. direct computation with *antaRNA*

Motivation and Setup: A brute force inverse folding approach to get sequences for a given target GC-value can be implemented by a 'sample and filter' pipeline with existing non-GC-optimizing tools. Therein, sequences are sampled and each is tested if it shows the targeted GC-content. Thus, the question arises how performant is such an approach in comparison to dedicated GC-optimization as done by *antaRNA*? Randomly we selected three structures and their constraints from the Rfam dataset. The used Rfam families RF00480, RF00007, RF00563 represent three different length categories, where $L1 = 1 \leq x < 100, L2 = 100 \leq x < 200, L3 = 200 \leq x < 300$. The presented test is exemplarily and does not cover a full range of all programs.

In order to investigate this, sample and filter pipelines based on the tools *NUPACK*, *ERD* and *RNAinverse* have been compared against *antaRNA*, using the aforementioned structures and their constraints. For each pipeline 1,000 sequences have been sampled and according timings have been recorded. Given a target GC-value (0.2, 0.3,.., 0.8), we computed for a given tool the averaged accumulated runtime in the sample set to find the first 10 sequences that show the targeted GC-content. If less than 10 sequences for a given target GC-value are found, the averaging was adopted. Note, for most target GC-values no sequence was found within the sample set, such that no runtime estimates are provided. For comparison, *antaRNA* was executed 100 times per target GC value and the same average runtime estimation scheme was applied. The results are visualized in Fig. S3.

Results and Conclusion: For all tools, runtime increases with increasing length and structure complexity of the test case examples (L1-L3). The 'sample and filter' pipelines often fail with increased length and complexity (target GC-value $\neq$ 50%) of the design task. With increasing difficulty only a target GC content of 50-60% was accessible. In contrast, *antaRNA* always produced according sequences and the runtime always superseeds the sample and filter pipelines. Notably, runtime is also correlated with length but less with the targeted GC-value. The more extreme target GC-values seem to increase the difficulty of the design task.

This experiment demonstrates the need for dedicated methods like *antaRNA* when designing sequences for a specific target GC-value. Sample and Filter pipelines show only very limited capability to produce according sequences in a reasonable amount of time, if at all.

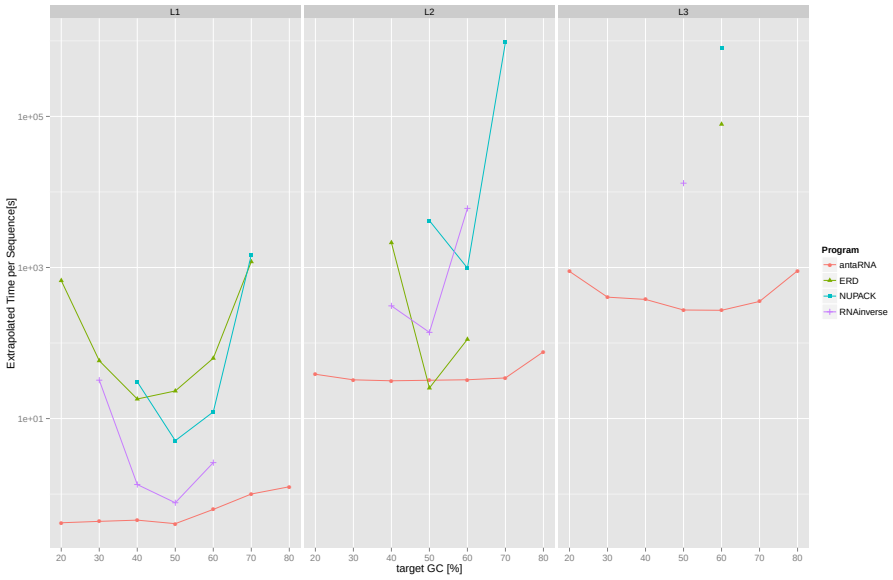

Figure S3: **Runtime comparison** of a Sample and Filter pipeline using *NUPACK*, *ERD* or *RNAinverse* versus direct *antaRNA* computations. Note, the sample and filter pipelines often miss to produce any sequence (among a sample of 1,000) for some target GC-values.

S7 Time comparison of *IncaRNAtion* and *antaRNA*

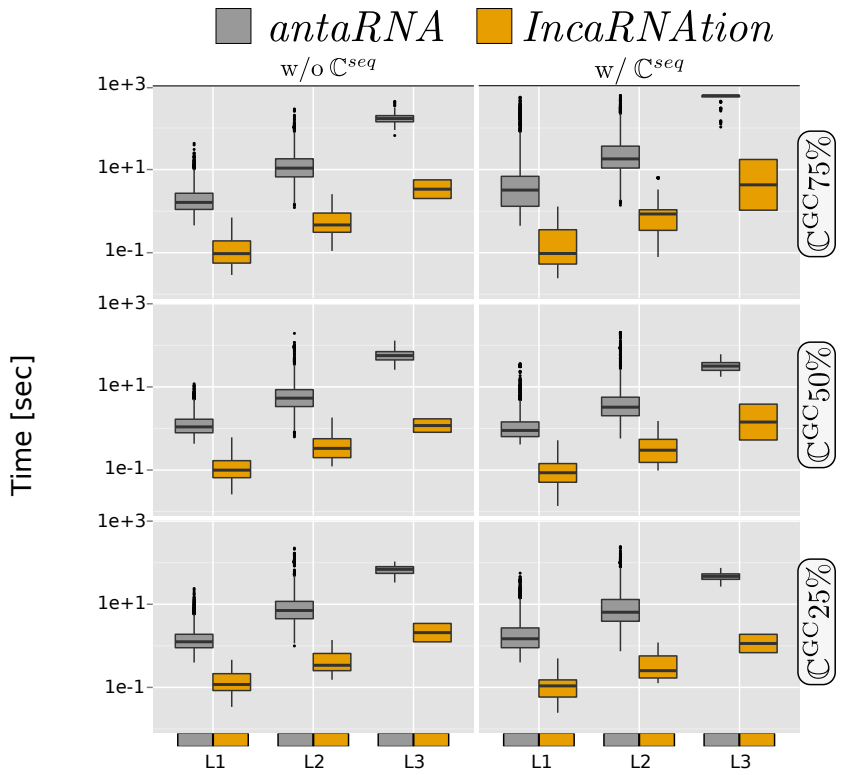

Figure S4: **Timing comparison** The presented results comprise the timings of runs of *antaRNA* (gray), *IncaRNAtion*(yellow). The distributions show different different performance towards different objective target GC values. The runs have been executed under the derived Rfam constraint test dataset . *RNAiFold*was not included into this comparison, since the computations were made externally on different machines. The runs have been performed with and without the respective Rfam sequence constraints $\mathbb{C}^{seq}$. Different target GC-content value $\mathbb{C}^{\text{GC}}$ have been tested (top row 75%, middle row 50%, bottom row 25%). For each constraint set, 100 sequences have been generated targeting the respective GC-content. The datasets have been split according to sequence length categories (L1:1-100; L2: 101-200; L3:201-300).

The measurements refer to the used Rfam dataset and consider the respective measured times. For each length category (L1:$\leq$100; L2:101-200; L3:201-300) the used seconds are plotted. The maximal allowed runtime per instance was set to 10 minutes for *antaRNA* and unlimited time consumption for *IncaRNAtion*. *IncaRNAtion* runs are in general faster than those of *antaRNA*, but as shown in the main paper, the quality of the produced entities is not as good as those of *antaRNA*.

References

[1] I. L. Hofacker, W. Fontana, P. F. Stadler, S. Bonhoeffer, M. Tacker, and P. Schuster, "Fast folding and comparison of RNA secondary structures," *Monatshefte Chemie*, vol. 125, pp. 167–188, 1994.

[2] R. Lorenz, S. H. Bernhart, C. Höner Zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker, "ViennaRNA Package 2.0," *Algorithms Mol Biol*, vol. 6, p. 26, 2011.

[3] D. H. Turner and D. H. Mathews, "NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure," *Nucleic Acids Res.*, vol. 38, no. suppl 1, pp. D280–D282, 2010.

[4] K. Darty, A. Denise, and Y. Ponty, "VARNA: Interactive drawing and editing of the RNA secondary structure," *Bioinformatics*, 2009.