Advanced design of structural RNAs using RNARedPrint

Yann Ponty[1], Stefan Hammer[2,3], Hua-Ting Yao[1,4], and Sebastian Will[1,3]

[1] LIX, CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
[2] Department of Theoretical Chemistry, University of Vienna, Austria
[3] Bioinformatics Group, Interdisciplinary Center of Bioinformatics, University of Leipzig, Germany
[4] School of Computer Science, McGill University, Montreal, Canada

Abstract. RNA design addresses the need to build novel RNAs, e.g. for biotechnological applications in synthetic biology, equipped with desired functional properties. This chapter describes how to use the software RNARedPrint for the *de novo* rational design of RNA sequences adopting one or several desired secondary structures. Depending on the application, these structures could represent alternate configurations or kinetic pathways. The software makes such design convenient and sufficiently fast for practical routine, where it even overcomes notorious problems in the application of RNA design, e.g. it maintains realistic GC content.

Keywords: RNA design; Kinetic landscapes; Riboswitches

1 Introduction

RNA design targets a wide diversity of biological functions and, as such, encompasses of wide array of computational tasks. Two dominant paradigms dominate current computational approaches:

- **Negative design** focusses on the **specificity** of produced sequences for "design targets". This can comprise the attempt to avoid functions, interactions, structures, or other properties that differ from the targeted ones. Such tasks correspond to inverse combinatorial problems, and can be computationally intractable (NP-hard) even when their direct version can be optimized in polynomial time [1].
  In the context of structural RNA design, negative design is usually referred to as the **inverse folding** problem [9], and consists in producing nucleotide sequences that uniquely folds into the target structure with respect to the Minimum Free Energy (MFE) criterion. Variants of the inverse folding problem consider the minimization of various notions of **defects** [5], notably including the **ensemble defect** [12], the expected base-pair distance between the target and a random structure in the Boltzmann-Gibbs distribution.
  An example of RNA design for a single target structure is provided in Figure 1, showing the results from our first running example of the Methods

section. As natural extension of single structure design, we will moreover discuss RNA design for multiple structural targets (cf. Fig. 3).

– **Positive design** can be loosely defined as focusing on the **propensity** of produced RNAs to achieve a certain function. In a structural context, positive design usually involves generating one or several sequences having good affinity (i.e. high stability ≈ low free-energy) for a targteted structure. Functionally, design constraints will also include the presence/absence of sequences motifs, a controlled affinity towards interactions with molecules, or a control of composition biases, such as the GC-content [10].
Recently, the objectives of positive design have been extended to include sampling of sequences in a controlled distribution induced by the design objective [10, 7, 8]. Interestingly positive design approaches implementing a controlled sampling strategy can be used to empirically tackle negative design objectives, by coupling a random generation of sequences, filtered to only retain good candidates for the negative design. Indeed, as shown in Figure 2, negative design objectives, such that the Boltzmann probability of the target structure, or the distance of the MFE to the target, tend to correlate well with positive design objectives, such as the free energy of the target structure.

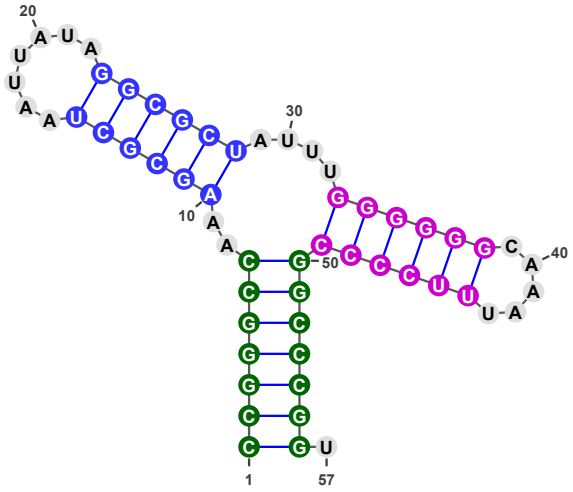
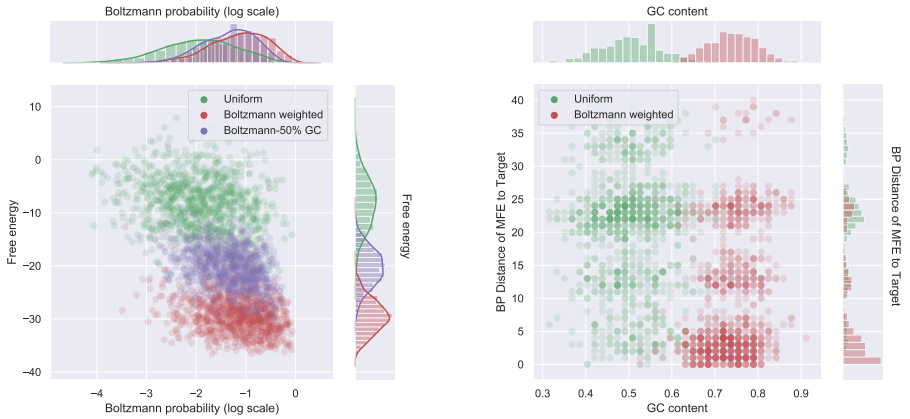


Fig. 1. The target RNA structure of our running example for single-target design, together with the finally designed sequence. We show its representation as as 2D plot (top, rendered using `VARNA` [3]) and dot-bracket string (below). The latter represents base pairs by balanced parentheses.



a) Boltzmann probability *vs* Free-energy of target

b) GC content *vs* distance of MFE to target structure.

Fig. 2. Negative design can be addressed by positive design in Boltzmann-weighted distributions on sequences. Distribution of Boltzmann probability, free-energy of target (a), GC content and distance of MFE to target structure (b). For a target structure `(((((((...(((((.....)))))).....(((((.....)))))))))))..`, 1 000 sequences were generated, either uniformly or in the Boltzmann distribution.

Applications of RNA design. From a molecular biology perspective, designing RNAs represents the **ultimate stress-test** of our understanding of how RNA folds and acts on its environment. In this setting, one designs RNA sequences expected to fold into a predefined structure with respect to a folding prediction model. Synthesizing the resulting sequences, and using experimental methods to verify the actual adoption of the desired structure, one either validates the model or reveals some of its flaws, fueling and prioritizing further developments. Indeed, misfolding designed RNAs reveal gaps in our energy models and descriptors of the conformation spaces used by predictive algorithms. A similar strategy can be more generally used to test functional hypotheses involving the structure of RNA.

Molecular design also represents one of the primitives of **synthetic biology**, and RNAs have been used in multiple roles, notoriously including biosensors [6], regulators, and nano-materials. Some naturally-occurring RNAs are sufficiently stable to fold in a modular fashion, enabling a *copy/paste approach* that simply combines existing RNAs into complete architectures. However, such a strategy is hindered, in an *in vivo* context, by the competition of artificial and endogenous RNAs, and by the intrinsic difficulty to produce orthogonal constructs over a limited number of available architectures. A rational design, coupled with an experimental filtering phase, is thus likely to represent the method of choice for future endeavors in RNA-based synthetic biology.

At a (primarily) sequence-based level, design is essential for future developments of **RNA-based therapeutics**. For instance, the recent discovery of

viable treatments based on RNA interference is fueled by an understanding of how small RNAs can interfere with selected messenger RNAs to activate or inhibit them. In this context, an optimization of the nucleotide sequence of small interfering RNAs, akin to a design task, is crucial to ensure its efficacy and selectivity, mitigating the risk of side-effects for the drugs. Similarly, the specificity of genetic contents targeted by CRISPR-based editing, an thus its limited toxicity, is ensured by a redesign of guide RNAs. More generally, the adoption of a stable structure is very often a prerequisite for the interaction of RNAs with selected proteins [4], and are therefore crucial for the functionality of designed RNAs in a cellular context.

Design also helps in the **search for homologous RNAs**. Indeed, in many RNA families, selective pressure mostly applies at the structure level. This aggravates the discovery of new occurrences of given RNA genes within the same organism (paralogs) or across available genomes (orthologs). If a structural model is known, and if the number of identified homologs is limited, a natural approach is to enrich homologs with sequences designed as to fold into the shared structure, in order to cover a larger proportion of the (neutral) sequence space.

Overview In this chapter, we describe how to install and apply RNARedPrint [8] to perform positive and negative RNA design. The method was designed to simultaneously account for the constraints induced by multiple RNA secondary structures. Its core capability is to generate sequences that achieve predefined thermodynamic stabilities for the target structures, while controlling the GC content. Consequently, we start by discussing design for single RNA structures and go on to the design of RNAs that fold into multiple structures. We demonstrate how to take advantage of RNARedPrint's versatility, for tackling a large variety of RNA design tasks.

2 Material

2.1 Installing RNARedPrint and Its Dependencies

RNARedPrint can be conveniently installed using the package manager Conda For Linux or macOS systems, we highly recommend this way of installing the software and provide detailed instructions below. For other systems or other types of installation, it is possible to install directly from the source code at `https://github.com/yannponty/RNARedPrint` following the there provided instructions. In this chapter, we describe the release 0.3 of the software.

Package manager Conda installation. You may skip this section if Conda is already installed and set. Otherwise, one can install Conda by installing Miniconda from `https://conda.io/en/latest/miniconda.html`. This could also be done from command line for Linux users

```
wget \
https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
sh Miniconda3-latest-Linux-x86\_64.sh
```

or if you have macOS, use

```
wget \
https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh
sh Miniconda3-latest-MacOSX-x86_64.sh
```

It is moreover convenient to set up several Conda channels (i.e., repositories for Conda packages), by

```
conda config --add channels defaults
conda config --add channels conda-forge
conda config --add channels bioconda
```

We require the bioconda channel, a Conda channel dedicated to bionformatics software, which contains the RNARedPrint package.

RNARedPrint installation. After following the instructions above, RNARedPrint can be installed using the command

```
conda install rnaredprint
```

This will install the tool `RNARedPrint` and three complementary Python scripts, `design-energyshift.py`, `design-multistate.py`, and `calcprobs.py`. Make sure to activate the Conda environment before calling the programs:

```
conda activate
```

3 Methods

3.1 General Usage

For typical advanced design tasks, our software package provides easy-to-use Python scripts as high-level front-ends to the computational engine of the software, implemented in the program `RNARedPrint`. It implements the package's core functionality of sampling RNA sequences from a specific distribution controlled by multiple targets.

The script `design-energyshift.py` generates RNA sequences with highly specific GC content and energies for given secondary structures. For this purpose, it implements a multi-dimensional Boltzmann sampling strategy on top of `RNARedPrint`. Our second script `desing-multistate.py` samples start structures for further optimization according to negative design objectives like probability or ensemble defect. Such optimization can be performed, e.g., using the

classical tool RNAinverse [9] (included in the already installed Vienna RNA package) or the recent software RNABluePrint [7]. Only the latter is equipped to handle the general and more complex case of multi-target design. To directly select sequences with high probabilities of the target sequences, a typical negative design criterion, we provide the script `calcprobs.py`. In the following, we provide concrete examples of the usage of all these tools in the two scenarios: first, the design for a single target structure; and second, the design for multiple structural targets. All these tools can be invoked from the command line of a terminal. The command line interface lets the user flexibly control the various options of the tools and, for advanced usage, enables integrating them into larger workflows.

The tool `RNARedPrint`. This work horse of our software supports generating sequences from a multi-dimensional Boltzmann distribution controlled by weights of the GC content and for the energies of one or several RNA secondary structures.

Our high-level scripts that can be used to address typical design scenarios, rely on this tool performing the main computation. Here, the tool serves us as quick test of the installation. Make sure that `RNARedPrint` is installed and found in your search path by running

```
RNARedPrint --version
```

If the software was installed successful (if you install via Conda as described above, do not forget to activate Conda), it reports its version in the form

```
RNARedPrint 0.3
```

Generally, the tools of the package provide usage information via the command line argument `--help`. Please have a look at the output of the command

```
RNARedPrint --help
```

to get a brief, complete overview of the usage of the tool. We will demonstrate the main usage in the subsequent sections.

3.2 Designing for a Single Target Structure

The most common case of RNA sequence design asks for sequences that fold well into a single RNA structure. While RNARedPrint is as well prepared to handle the more challenging task of multi-structure design, we will start with this simpler case to introduce the software. Thus, let us demonstrate how to design RNA sequences for our example target structure illustrated in Figure 1.

Target secondary structures are given as a "dot-bracket string", where each pair of balanced parentheses describes a base pair. This is the typical secondary

structure linearization prominently popularized by the Vienna RNA package. We will use this representation throughout our explications and in the input and output of our design tools.

A popular approach to RNA sequence design is to efficiently generate start sequences, also called seeds, that are subsequently optimized using local search strategies. Our software allows to design 'good' seed sequences, where we have strong control over the GC content of the sequences and their energies for the target structure. Recall that the design of such start sequences is a case of *positive design*.

Positive design for a single target structure with RNARedPrint. Let us start by calling `RNARedPrint` from the command line (again, from the installation directory) as

```
RNARedPrint --num 4 --weights 5 --gcw 0.5 \
  "(((((((...(((((.......)))))).....(((((.......)))))))))))."
```

With these arguments, we ask the tool to sample 4 sequences S (`'--num'`) with probabilities proportional to

$$5^{-E_{\mathrm{bp}}(S)} \cdot 0.5^{\#GC(S)}, \tag{1}$$

where $E_{bp}(S)$ denotes the energy of the sequence and the target structure in the simple base pair energy model (cf. Section 3.3) and $\#GC(S)$ denotes the number of bases G and C in the sequence. Note how the weights are set by options `--gcw` and `--weights` and affect the distribution due to Equation 1. Without such options, the tool would use default weights of 1.

A typical example of the output produced by the tool is:

```
CGUCCACGAUACCCGUACGCUAUGGGUGGUGUGCCCCUCAAAAUGGGGCGUGGAUGA GC=0.59 E1=-22.87
CCGCGUCUGUAGCCCCGAUAACGGGUUUAUACAGGUUAUAUUAUGGCUUGACGUGGC GC=0.51 E1=-19.22
GUUGGUUUUGGUCUACGAAGCGUAGACAGGUACCCCAUAUGUAGUGGGGAACCAAUA GC=0.48 E1=-16.52
GGGGGCGAACGUCCAUAUCGGAUGGACUAUUUGCCCGUCUUAAGCGGGCCGCCUUUU GC=0.57 E1=-23.11
```

By changing the weights, the distributions and thereby the means of the energy and the GC content can be changed. Increasing the energy weight causes the generation of sequences with better energy. In tendency, due to RNA thermodynamics, these sequences have higher GC content. At the same time lowering the GC-weight (`--gcw`) allows counteracting and keeping the GC content controlled. To observe this effect, e.g., compare the outputs of the following two calls

```
RNARedPrint --num 10 --weights 20 --gcw 0.5 \
  "(((((((...(((((.......)))))).....(((((.......)))))))))))."
```

and

```
RNARedPrint --num 10 --weights 20 --gcw 0.2 \
  "(((((((...(((((.......)))))).....(((((.......)))))))))))."
```

Targeting specific GC content and free energy. Playing around with different weights for `RNARedPrint` in an attempt to target specific (mean) GC content or specific energies quickly reveals that this is not at all trivial, since the different targets are not independent. For this purpose, we provide the script `design-energyshift.py`, which solves the optimization problem of finding weights, such that specific energies and GC content are targeted by multi-dimensional Boltzmann sampling.

Notably, the script supports targeting specific energies in the accurate *Turner energy model* [11], whereas the tool `RNARedPrint` samples based on energies in a simple model (by default, the base pair energy model).

This script allows specifying targets, e.g. GC content of 60 percent and (Turner) energy of -28 kcal/mol, for the designed sequences in a call like

```
design-energyshift.py --num 4 --gc 0.6 -e=-28 \
  <<<"(((((((...(((((.......)))))).....(((((.......)))))))))))."
```

By this call, the script produces output similar to

```
CCGGGCCAAAGCGCUAAUUAUAGGCGCUAUUUGGGGGGCAAAUUUCCCCGGCCCGGU GC=0.59 E1=-28.70
CGCCAGCUGCCCUCUAUACAUUAGAGGAAUUUGCGCGUAUCCGUCGCGCGCUGGCGA GC=0.59 E1=-27.00
GGCCCGUUCUGCGCCGAUAUUAGGCGCUCACACGCCGUGUUAGUCGGUGAUGGGCCA GC=0.62 E1=-27.70
CCGCCACAUAGCGCUCGUGAAUAGCGCAAGGAGCGCCAUCAAAAGGCGCGUGGCGGA GC=0.62 E1=-28.90
```

Negative design for a single target. The RNA sequences generated by the script `design-energyshift.py`, while targeting low energies, are typically good designs for the target structure even according to negative design criteria like MFE or even probability and ensemble defect (cf. Fig. 2). To quickly test this for concrete examples, one can fold the designs using `RNAfold`. Let us demonstrate this for the first of the above designs.

```
RNAfold \
  <<<CCGGGCCAAAGCGCUAAUUAUAGGCGCUAUUUGGGGGGCAAAUUUCCCCGGCCCGGU
```

RNAfold reports the MFE structure together with additional information about the RNA structure ensemble.

```
CCGGGCCAAAGCGCUAAUUAUAGGCGCUAUUUGGGGGGCAAAUUUCCCCGGCCCGGU
(((((((..((((((.......))))))....((((((.....)))))))))))). (-30.80)
```

We observe that the distance between MFE and target structure (i.e., the MFE defect) is only two base pairs; the structures look almost identical. The next design even yields optimal MFE defect of zero. This suggests that negative design can be performed rather effectively by screening through a series of (positive) designs by RNARedPrint. Even more effective is the use of such designs of start structures in a local optimization according to negative design objectives. Such optimization is possible using tools like RNAinverse or RNABluePrint.

To apply the former, we feed it with our target structure and, e.g. our first designed sequence

```
echo -e \
  "(((((((...(((((.......)))))).....(((((.......)))))))))))).\n\
  CCGGGCCAAAGCGCUAAUUAUAGGCGCUAUUUGGGGGGCAAAUUUCCCCGGCCCGGU"\
  | RNAinverse -Fp
```

The given option causes RNAinverse to optimize the probability of the target structure. On this input, RNAinverse succeeds quickly, returning a very good design for the target structure.

```
UUAGAUCAAAUAGGAGUCGAUGUCCUGGGGGUACGUGAAACAAGCACGUGAUUUAGU   23
GGCCGGCGAAGGGGCGAAUAAAGCCCCAAAAACCCGCGAAAAAAGCGGGGCCGGCCA   47  (0.989619)
```

The finally designed sequence (last row) forms the target structure with a probability of 0.99. Notably, the GC content of the design—which used to be hard to control by classic design methods—is fairly close to the start sequence.

For harder designs than our running example, these observations suggest an automated two step approach to negative design. First, a set of start sequences is systematically generated using positive design. Second, these sequences are optimized, e.g. using RNAinverse, as shown above. Similar strategies have been discussed in the literature, e.g. by the early approach INFO-RNA [2] or using the tool IncARNation [10]. The latter is implementing the exact same ideas of Boltzmann sampling for positive RNA design, including the control of GC content. Due to our software RNARedPrint, which easily covers single-structure design as a special case, this approach to single target design got even more accessible and flexible.

3.3 Multi-Target Design

Positive design by RNARedPrint. Let us demonstrate the use of RNARedPrint for multi-target design by running the tool for a small instance (sequence length 40) with three target structures. As preparation we create the file `targets.txt` containing the target structures (one per line)

```
((((((((....))))))))((((((((....))))))))
((((((((..((((((((...))))))))...))))))))
((((((((((((((((...))))))))))))))))...
```

Then, we call the tool by

```
RNARedPrint --weights 1,2,5 --gcw 0.5 --num 5 $(cat targets.txt)
```

This call specifies three target secondary structures and asks for five sequences that have the same length as the structures. The call sets the weights for the three target structures (respectively 1,2, and 5) and the weight for the GC content. The tool automatically seeds its random number generator, such that repeated calls produce different output. The output should look like

```
(((((((....)))))))(((((((....)))))))
(((((((..(((((((...)))))))...)))))))
((((((((((((((...))))))))))))))...
GGAGGGGGCCCCCUCCUUUUGGGGGGGGCCCCUUCCUUC GC=0.73 E1=-18.29 E2=-21.07 E3=-28.83
GAGGGGGGCUCCCCCUCUUUGGGGGGGGCCCCCCUUUUC GC=0.73 E1=-19.51 E2=-22.29 E3=-28.49
GGGGGGGGCUCCCCCUCUCCGGGGGGGGCCCCCCCCUCC GC=0.86 E1=-25.96 E2=-26.30 E3=-31.29
GGGGGGGGUCCUUCCCUUUCGGGGGGGGUCCCCCCCCU GC=0.77 E1=-23.86 E2=-23.86 E3=-27.98
GGGGGGGGCCCCCCCCCCUUGGGGGGGGCCUCCCCCUU GC=0.84 E1=-24.74 E2=-27.18 E3=-32.51
```

As in the single-target case, decreasing (or increasing) the weight '`--gcw`' will produce sequences with, in tendency, lower (or, respectively, higher) GC content. Similarily, decreasing (or increasing) the weight of some secondary structure, generates sequences with higher (or lower, i.e. better) energy. Thus, changing the weights allows shifting the distribution means of the different features of the generated sequences (i.e., the GC content and energies for the different structures).

Targeting multiple specific energies and GC content. Targeting very specific values of several features, again requires the support of advanced automatized methods—here, this is even more important than for single target design. Again, the corresponding hard optimization problem is solved performing multidimensional Boltzmann sampling by the script `design-energyshift.py`.

In generalization of the single-target case, the script allows generating sequences with sharply defined values of the multiple targeted features. For example, we can ask for sequences with respective Turner energies of -18,-22, and -20 for the three previously introduced target structures. At the same time, we want aim at GC contents of about 65 percent.

To obtain five such sequences, one calls the script as

```
design-energyshift.py -i targets.txt --num 5 \
  --gc 0.65 --energies="-18,-22,-20"
```

which produces five designs like

```
GGGGGGGGUUCCUCUCUCUUGGAGGGGGGUUUUCUCCUCC GC=0.64 E1=-17.30 E2=-22.20 E3=-19.40
GGGGGGGGCUUCCUUUCCUUGGGGGGGGGUUUUCCCUCCC GC=0.68 E1=-19.00 E2=-21.90 E3=-20.40
GGGGGGGAUCCCUUCUCCUUGGGGGGGGGUUUUCUUCCCC GC=0.66 E1=-17.60 E2=-22.30 E3=-20.50
GGGAGGGGUUCCCUUCUCUUGGGGGGGGGUCUUUCCUCCC GC=0.66 E1=-17.20 E2=-22.60 E3=-19.70
GGGGGGAGUUCCCUUUCCUUGGGGGGGGGCUUCUUUCCCC GC=0.66 E1=-17.30 E2=-22.10 E3=-20.60
```

By default, the script tolerates deviations from the targets by as much as 1 kcal/mol. We can define the tolerance, e.g. more strictly as ±0.5 kcal/mol using the additional argument `--tolerance 0.5`. Similarly, the tolerance of the GC content can be selected and the operation of the tool can be fine-tuned in various other ways. When called with option `--help` the script provides a full overview of available arguments.

Aiming at negative design criteria. As we already discussed before, a major motiviation for positive design is to produce sequences that perform well

according to negative design criteria. From a sample of such sequences, one can either already satisfy negative design requirements or find good start sequences for further refinement, e.g. by stochastic local search.

A typical requirement for good designs in the case of single target design can be expressed in slight simplification as "good stability, avoiding extreme GC contents". In the multi-structure case, we additionally aim at specific relations between the energies. This is possible with our tool `design-energyshift.py` by targeting energies for each structure.

The script `design-multistate.py` was written to design sequences, where all structures have highly similar energies, which is a common objective in RNA design. Let us demonstrate this for the previously introduced three target structures (in file `targets.txt`).

To produce 5 designs targeting similar energy for all three targets, while controlling the GC content at about 0.65, the script is called as

```
design-multistate.py -i targets.txt --num 5 --gc 0.65
```

It returns the designs together with their GC content and Turner energies:

```
GGGGGGGGUCUUCCCUUCUUAGGGGGGGGUUCUCCCCCCCU GC=0.68 E1=-24.20 E2=-22.90 E3=-24.00
AGGGGGGGUUCUCCCCCUCUUAGGGGGGAGUUUUCCCUCUU GC=0.59 E1=-22.30 E2=-23.80 E3=-22.70
GGGGGGGGUCUCCCUUUCUCGGAGGGGGGUUUUCUCCCUCC GC=0.68 E1=-22.80 E2=-22.40 E3=-22.30
GGGGGGGAUUUCUCCCUCUCGGGGGGGGAUUUUCCUCUCC GC=0.64 E1=-23.10 E2=-23.60 E3=-22.70
GGGGGGGAUUCUUCCCUUCUAGGGGGGAGUUUUUCCCCCU GC=0.59 E1=-23.70 E2=-23.10 E3=-23.00
```

Similar to the script `design-energyshift.py`, further options provide additional control over the script operation, and an overview of arguments can be produced by running the script with the `--help` option.

Negative design due to positive design. Sequences as designed above can be used as start sequences for refinement due to negative design criteria, which cannot be directly targeted by the sampling engine.

Refinement by local optimization can be performed using third-party software. For the case of single-target design, we demonstrated the use of RNAinverse, as the 'classic' tool for this purpose. However, different to the single-target case, corresponding software for multi-structure design is rare. One good choice would be the design software RNABluePrint.

In many cases good solutions to negative design criteria can be found by screening a larger number of positive designs. This mainly requires us to re-evaluate sequences as generated by using our scripts according to negative criteria. We demonstrate this for the criterion of accumulated probability of the target structures. Figure 3 shows good negative designs for our three target sequences (as well as, for comparison, one single-target design) that we obtained following this strategy.

For the purpose of re-evaluating the positive designs, we provide the script `calcprobs.py`. It can be used to post-process the output of our design scripts. For each sequence in the ouptut it annotates the corresponding output line. For example, when reading the lines

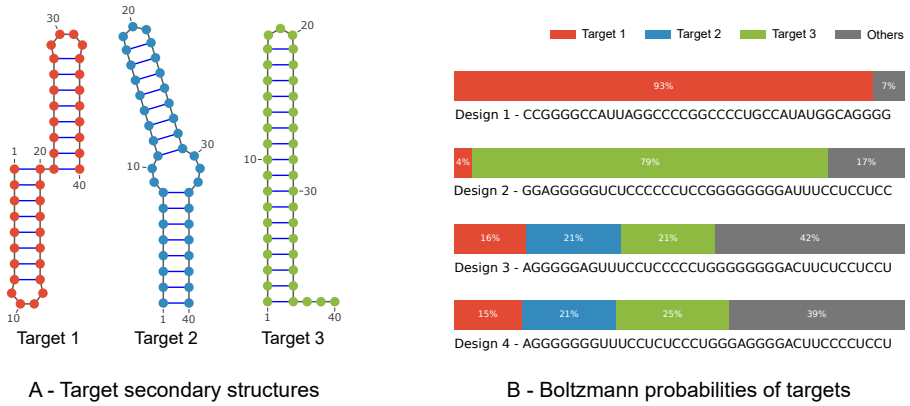


Fig. 3. Negative design for three target structures. For three target structure, inducing disjoint sets of base pairs (A – left), sequences are designed as to simultaneously optimize the Boltzmann probabilities (B – right) of the first (Design 1), third (Design 2) and second (Designs 3 and 4) target structures. For the two latter sequences, the probabilities of the three targets were designed to be comparable, *e.g.* in order to address the need for a switching behavior at the thermodynamic equilibrium.

```
AGGGGGAGUUUCCUCCCCCUGGGGGGGGACUUCUCCUCCU GC=0.66 E1=-26.90 E2=-27.10 E3=-27.10
AGGGGGGGUUUCCUCUCCCUGGGAGGGGACUUCCCCUCCU GC=0.66 E1=-26.90 E2=-27.10 E3=-27.20
```

the script calculates the minimum free energy (MFE) and the ensemble energy of the sequences (EE). Then it computes the probabilities of each target for the sequences (Pi) and their sum (Psum). Consequently, it extends the lines to report this information by the respective strings

```
MFE=-27.10 EE=-28.05 P1=0.16 P2=0.21 P3=0.21 Psum=0.58
MFE=-27.20 EE=-28.05 P1=0.15 P2=0.21 P3=0.25 Psum=0.62
```

The above two lines correspond to the two last designs of Fig. 3. They were obtained as best negative designs, according to `Psum`, after generating and annotating 1000 samples by

```
design-multistate.py -i targets.txt --num 1000 \
  --gc 0.7 --energy="-27" --tolerance=0.2 \
  | calcprobs.py  -i targets.txt
```

where `calcprobs.py` is used in a pipeline with the design script.

Notes

Boltzmann distributions as generated by RNARedPrint The fundamental computational task in the RNARedPrint is the generation of Boltzmann samples

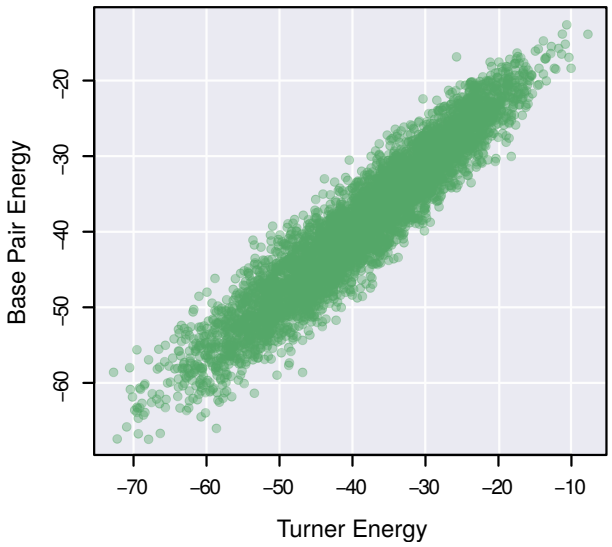


Fig. 4. Strong correlation of 0.95 between energies in the simple base pair model and Turner energies reported by the Vienna RNA package; determined for 5 000 uniform random RNA sequences and their MFE structures.

of sequences. This allows controlling the frequencies of feature values, namely the GC content and the energies of the target structures (in a simple energy model). The distribution of the different features is furthermore controlled by weights for each of the features. Consequently, each sequence is generated with a probability that is proportional to the sum of "feature weight to the power of the feature value" over all features. In Eq. 1 we gave an example for the simple case of a single structure.

Uniform sampling of sequences. Uniform sampling is a special case of Boltzmann sampling. If we set all weights to 1, which is the default of RNARedPrint, the it produces a uniform sample of the sequences. Effectively this turns off the influence of feature values (since 1^x equals 1, such that the proportionaly factor is constant for all sequences).

Simple energy model as efficient proxy for the Turner model. For efficient sampling in RNARedPrint, we utilize simple energy models that approximate the highly accurate nearest neighbor Turner RNA energy model. For this purpose, we trained parameters [8] such that the energies even in the base pair model show good linear correlation to the Turner energies (Fig. 4). The base pair model evaluates energies as a sum of energy terms for each base pair that depend only on the type of the base pair ("AU", "CG" or "GU") and furthermore

distinguishes stacked and non-stacked base pairs. For details, see [8], which as well discusses the slightly more complex stacking energy model.

The strong correlation between energies in the two models, allows us to much more efficiently target energies in the simple model for the purpose of finally targeting energies in the realistic Turner model. Utilizing the simple model as proxy for the realistic model is indeed a crucial aspect for the viability of our method.

References

1. Édouard Bonnet, Paweł Rzażewski, and Florian Sikora. Designing RNA secondary structures is hard. In Benjamin J. Raphael, editor, *Research in Computational Molecular Biology - 22nd Annual International Conference, RECOMB 2018*, volume 10812 of *Lecture Notes in Computer Science*, pages 248–250, Paris, 2018. Springer.
2. Anke Busch and Rolf Backofen. INFO-RNA—a fast approach to inverse RNA folding. *Bioinformatics*, 22(15):1823–31, 2006.
3. Kévin Darty, Alain Denise, and Yann Ponty. VARNA: Interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, 25(15):1974–5, August 2009.
4. Natalia Sanchez de Groot, Alexandros Armaos, Ricardo Graña-Montes, Marion Alriquet, Giulia Calloni, R. Martin Vabulas, and Gian Gaetano Tartaglia. RNA structure drives interaction with proteins. *Nature Communications*, 10(1), July 2019.
5. Robert M. Dirks, Milo Lin, Erik Winfree, and Niles A. Pierce. Paradigms for computational nucleic acid design. *Nucleic Acids Research*, 32(4):1392–1403, 2004.
6. Sven Findeiß, Maja Etzel, Sebastian Will, Mario Mörl, and Peter F Stadler. Design of artificial riboswitches as biosensors. *Sensors (Basel, Switzerland)*, 17(9):E1990, August 2017.
7. Stefan Hammer, Birgit Tschiatschek, Christoph Flamm, Ivo L Hofacker, and Sven Findeiß. RNAblueprint: flexible multiple target nucleic acid sequence design. *Bioinformatics (Oxford, England)*, 33:2850–2858, September 2017.
8. Stefan Hammer, Wei Wang, Sebastian Will, and Yann Ponty. Fixed-parameter tractable sampling for RNA design with multiple target structures. *BMC bioinformatics*, 20(1):209, 2019.
9. Ivo L Hofacker, Walter Fontana, Peter F Stadler, L Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie/Chemical Monthly*, 125(2):167–188, 1994.
10. Vladimir Reinharz, Yann Ponty, and Jérôme Waldispühl. A weighted sampling algorithm for the design of RNA sequences with targeted secondary structure and nucleotide distribution. *Bioinformatics*, 29(13):i308–i315, 2013.
11. Douglas H Turner and David H Mathews. NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic acids research*, 38(Database issue):D280–D282, January 2010.
12. Joseph N Zadeh, Brian R Wolfe, and Niles A Pierce. Nucleic acid sequence design via efficient ensemble defect optimization. *Journal of Computational Chemistry*, 32(3):439–52, 2011.