In silico design of a ligand triggered RNA switch

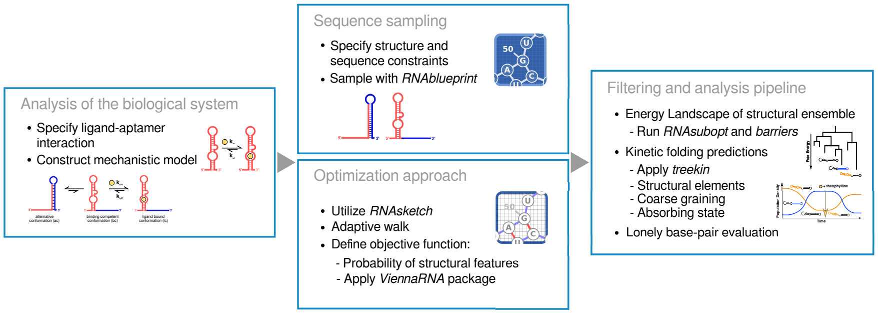
Stefan Hammer[1] and Sven Findeiß[1,⋆]

[1]Bioinformatics, Institute of Computer Science, and Interdisciplinary Center for
Bioinformatics, Leipzig University, Härtelstraße 16–18, 04107 Leipzig, Germany
⋆ *sven@bioinf.uni-leipzig.de*

January 15, 2019

Abstract

This contribution sketches a work flow to design an RNA switch that is able to adapt two structural conformations in a ligand-dependent way.
A well characterized RNA aptamer, i. e., knowing its K_d and adaptive structural features, is an essential ingredient of the described design process. We exemplify the principles using the well-known theophylline aptamer throughout this work. The aptamer in its ligand-binding competent structure represents one structural conformation of the switch while an alternative fold that disrupts the binding-competent structure forms the other conformation. To keep it simple we do not incorporate any regulatory mechanism to control transcription or translation.
We elucidate a commonly used design process by explicitly dissecting and explaining the necessary steps in detail. We developed a novel objective function which specifies the mechanistics of this simple, ligand-triggered riboswitch and describe an extensive *in silico* analysis pipeline to evaluate important kinetic properties of the designed sequences. This protocol and the developed software can be easily extended or adapted to fit novel design scenarios and thus can serve as a template for future needs.



Analysis of the biological system

- Specify ligand-aptamer interaction
- Construct mechanistic model

Sequence sampling

- Specify structure and sequence constraints
- Sample with *RNAblueprint*

Optimization approach

- Utilize *RNAsketch*
- Adaptive walk
- Define objective function:
  - Probability of structural features
  - Apply *ViennaRNA* package

Filtering and analysis pipeline

- Energy Landscape of structural ensemble
  - Run *RNAsubopt* and *barriers*
- Kinetic folding predictions
  - Apply *treekin*
  - Structural elements
  - Coarse graining
  - Absorbing state
- Lonely base-pair evaluation

k_on
k_off
5'
3'
alternative conformation (ac)
binding competent conformation (bc)
ligand bound conformation (lc)
50
U
G
A
C
Free Energy
Population Density
Time
+ theophylline

Keywords: synthetic biology, RNA design, inverse folding, multi-state design

Introduction

An *in silico* design process can in general be split into three parts: i) design idea, ii) the computational sequence generation and iii) the *in silico* analysis of the resulting sequences. The following sections describe each step in detail and the applied software is listed in Table 1. If you want to go through this protocol step by step we recommend to first download and install all tools and continue reading afterwards.

1 Idea

At the very beginning of a design process an idea of the RNA function to be implemented is mandatory. Due to RNAs close structure to function relationship, it is possible to draft a two dimensional structure which already infers function to some extend. We aim to design an RNA switch that is able to fold into two structural conformations in a ligand dependent way. Therefore, a well characterized RNA aptamer, i.e. knowing its K_d and adaptive structural features, is an essential component of the intended design. We exemplify the principles using the well known theophylline aptamer [6, 7] throughout this protocol.
The aptamer in its ligand binding competent structure (`bc`) represents one structural conformation of the switch while an alternative fold (`ac`) that disrupts the ligand binding competent structure forms the other conformation, see Figure 1. As soon as the ligand is added to the system, the formation of `ac` will be inhibited and the binding competent conformation (`bc`) becomes favored as the ligand will immediately bind to the this conformation which gets therefore further stabilized into its ligand bound conformation (`lc`). To keep the design simple we do not incorporate any regulatory mechanism to control transcription or translation.

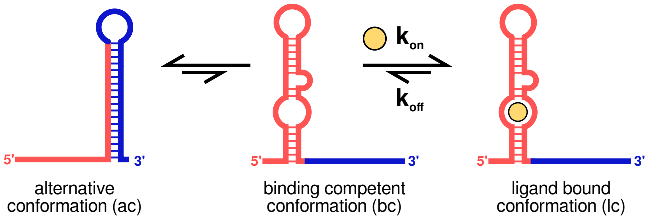

5'
3'
alternative conformation (ac)
5'
3'
binding competent conformation (bc)
k_{on}
k_{off}
5'
3'
ligand bound conformation (lc)

Figure 1: Graphical representation of the design idea. The system can be decomposed into two parts. Two conformations should dominate the structural ensemble of the designed RNA sequence in absence of the ligand. Depending on the design parameter the `alternative conformation (ac)` should be higher populated than the `binding competent (bc)` one. Refolding rates between the two structural conformations depend on the energy barrier that separates them. Upon ligand addition the `bc` gets trapped and the system should end up in the `ligand bound conformation (lc)`.

2 Computational sequence generation

To obtain a sequence with the requested characteristics, usually an optimization problem is formalized, where a generated **seed sequence** is **iteratively mutated** and evaluated with an **objective function** describing the desired biophysical properties of the system. The **optimization method** defines which mutations are kept and when to stop the iterative search. A variety of well-established optimization methods such as the Metropolis–Hastings algorithm [12, 4] or genetic algorithm based approaches [14] were used to find optimal solutions by traversing through the constrained solution space. We will use a simple gradient walk approach to find a solution for our design objective.

2.1 Sequences compatible to constraints

To generate a valid RNA sequence we can simply draw one out of four nucleotides for each position of the sequence. However, for your example we would end up with a solution space of $1.36 \cdot 10^{39}$ sequences where many of those are not even able to adopt our desired structures deduced from the sketch of the design idea:

```
design_input.txt
# alternative conformation:
.........................(((((((((((......)))))))))))...........
# binding competent conformation:
(((((...((((((((.....)))))...)))...))))).........................
# sequence constraint:
AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCANNNNNNNNNNNNNNNNNNNNNN
#<---       aptamer constraint       --->|
```

To be more efficient, we use an algorithms that generate sequences compatible to the given structural and sequence constraints. For single structural targets, such an algorithm needs to take care of the base-pairing rules for paired bases and can sample one out of four for unpaired positions. However, to generate sequences compatible to two or even multiple structures and additional constraints, this task becomes tricky. A dynamic programming algorithm based on graph-coloring was therefore developed by [5] and implemented in `RNAblueprint` [3].

```
$ RNAblueprint -v < design_input.txt
```

Run the command to retrieve the information on how many sequences exist (for the given example $1.34218 \cdot 10^8$) and ten sequences compatible with the constraints.

2.2 Objective Function

Each of these sequences can fold into the specified structures *in silico*, but these do not have to be thermodynamic stable or even the minimum free energy (MFE) structure. To ensure these important properties, an objective function scores each sequence according to a design goal.

2.2.1 Ligand binding model

The objective function describes the design mechanism split into the system (i) with ligand and (ii) without ligand. If the ligand is present we want to trap the RNA in its binding competent and thereby in the presumably bound conformation. In absence of the ligand the alternative conformation should dominate the ensemble of structural states.

To incorporate ligand binding into the objective, a model aware of the stabilizing contributions of the RNA–ligand dimerization is required. Thus, the recently implemented soft constraint framework of the ViennaRNA package [10] has been applied. Among other things, it allows to add an energy bonus to structural states that exhibit a certain motif.

When evaluating the structure ensemble of a given molecule containing the theophylline aptamer sequence, an energy bonus of $\Delta G = -9.22\,\mathrm{kcal\,mol^{-1}}$ is added to every secondary structure that contains the correctly folded binding pocket. This value is obtained from the relation $\Delta G = R \times T \times \ln K_d$ for the gas constant $R = 1.987\,17\,\mathrm{cal\,mol^{-1}\,K}$, the temperature $T = 310.15\,\mathrm{K}$, and the experimentally measured dissociation constant $K_d = 0.32\,\mathrm{\mu M}$ [6]. Using the example sequence and applying the --motif option of RNAfold, the MFE structure contains the binding-competent aptamer fold with a corrected energy value.

```
testing_example.txt
AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCAGUUGUUGAGGGGGCUCAAUGAC

$ cat testing_example.txt | RNAfold
  AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCAGUUGUUGAGGGGGCUCAAUGAC
  ................((((((((.(((((((((((......))))))))))).).)))).))) (-21.60)

$ cat testing_example.txt | RNAfold \
  --motif="GAUACCAG&CCCUUGGCAGC,(...((((&)...)))...),-9.22"

  AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCAGUUGUUGAGGGGGCUCAAUGAC
  (((((...((((((((.....)))))...)))...)))))..(((((((((....))))))))) (-30.32)
```

2.2.2 MFE defect

The distance of the actual minimum free energy (MFE) structure, e.g. predicted with RNAfold, to the structural constraint (the desired MFE structure) might be an intuitive evaluation of our generated sequences.

```
rnadistance_ac.txt
.........................(((((((((((......)))))))))))...........
................((((((((.(((((((((((......))))))))))).).)))).)))
rnadistance_bc.txt
(((((...((((((((.....)))))...)))...)))))........................
(((((...((((((((.....)))))...)))...)))))..(((((((((....)))))))))

$ RNAdistance --compare=f --distance=F < rnadistance_ac.txt
F: 16
$ RNAdistance --compare=f --distance=F < rnadistance_bc.txt
F: 18
```

We use `RNAdistance` to count the number of positions that have to be changed, which is 16 and 18 for the `ac` and `bc` target structure to the MFE structure without and with ligand, respectively. An objective function that minimizes this distance to both structures could be defined as

$$f(x) = D(\phi_{\mathrm{MFE}}, \phi_{\mathrm{ac}}) + D(\phi_{\mathrm{MFE,lig}}, \phi_{\mathrm{bc}})$$

where ϕ_{MFE} and $\phi_{\mathrm{MFE,lig}}$ are the MFE structures without and with ligand, and ϕ_{ac}, ϕ_{bc} are the two target structures. Obviously, the value tends towards zero for a perfect design. However, this objective does not take the ensemble of structures or any energy terms other than the MFE stucture prediction into account and thus is far from ideal.

2.2.3 Structure probability defect

The sequence–structure mapping is a one-to-many relation. Hence, one sequence can adapt a huge set of possible structures Φ called this sequence's structure ensemble. thus, it would be better to incorporate the probabilities of certain structures in the ensemble, such as the main stem of the alternative conformation.

Given a sequence x and a compatible structure ϕ, one can calculate the corresponding Gibbs free energy $G(x\,|\,\phi)$ using the nearest neighbor model [11, 15] by calling `RNAeval`. In the equilibrium, the Boltzmann weight $B(x\,|\,\phi) := \exp(-\frac{G(x|\phi)}{RT})$ of a structure ϕ is proportional to its probability. Summing over all structures of the ensemble gives rise to the partition function $Z(x) = \sum_{\phi\in\Phi} B(x\,|\,\phi)$ of x. The latter can also be derived from the ensemble free energy $G(x\,|\,\Phi)$ by calculating $Z(x) := \exp(-\frac{G(x|\Phi)}{RT})$. From that we can calculate the probability of ϕ with respect to the ensemble as

$$P(x\,|\,\phi) = \frac{B(x\,|\,\phi)}{Z} = \exp(-\frac{G(x|\phi) - G(x|\Phi)}{RT}).$$

```
ac.txt
AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCAGUUGUUGAGGGGGCUCAAUGAC
.........................(((((((((((......))))))))))).........
bc.txt
AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCAGUUGUUGAGGGGGCUCAAUGAC
(((((...((((((((.....)))))...)))...))))).......................

$ cat ac.txt | RNAeval
  AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCAGUUGUUGAGGGGGCUCAAUGAC
  .........................(((((((((((......))))))))))).......... (-15.40)

$ cat bc.txt | RNAeval
  AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCAGUUGUUGAGGGGGCUCAAUGAC
  (((((...((((((((.....)))))...)))...)))))....................... (-12.20)

$ cat ac.txt | RNAfold -p
  AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCAGUUGUUGAGGGGGCUCAAUGAC
  ................((((((((.(((((((((((......))))))))))).).)))).))) (-21.60)
  ..,,{....,.,,|||(({{(((|,(((((((((((......))))))}}))).},)))).))) [-22.75]
  ................((((((((.(((((((((((......))))))))))).).)))).))) {-21.60 d=14.58}
 frequency of mfe structure in ensemble 0.154902; ensemble diversity 21.09
```

Thus, for our example the probability of our alternative conformation `ac` and the ligand competent conformation `bc` can be calculated:

$$RT = 1.98717 * (273.15 + 37.0)/1000 \text{ kcal mol}^{-1}$$

$$P(x \mid \phi_{\text{ac}}) = \exp(-\frac{G(x|\phi_{\text{ac}}) - G(x|\Phi)}{RT}) = \exp(\frac{15.40 - 22.75}{0,61632}) = 6.625e - 06$$

$$P(x \mid \phi_{\text{bc}}) = \exp(\frac{12.20 - 22.75}{0,61632}) = 3.683e - 08$$

For the probability of the ligand bound state `lc`, we can run the same commands, however we need to add the `--motif` option for the ensemble free energy calculation and add the ligand energy contribution to the `bc` structure energy.

$$P(x \mid \phi_{\text{lc}}) = \exp(\frac{12.20 + 9.22 - 30.98}{0,61632}) = 1.837e - 07$$

In presence of the ligand, the probability of `lc` should be maximized. In contrast, `ac` should be highly populated in absence of the ligand. However, no ligand binding is possible if the RNA molecule exclusively adapts `ac` as only `bc` induces a high binding affinity of the ligand for the RNA molecule. It is therefore necessary to establish a balance between `ac` and `bc` where `bc` must always be present. We combined all these assumptions into the objective function

$$f(x) = P(x \mid \phi_{\texttt{lc}}) \cdot (1 - |a - P(x \mid \phi_{\texttt{ac}})|) \cdot (1 - |b - P(x \mid \phi_{\texttt{bc}})|) \qquad (1)$$

where $a, b \in (0, 1),\ a + b \leq 1$ are the target probabilities of the alternative conformation and binding-competent conformation, respectively. This function is maximized as the three multiplied terms tend to one. We set $a = 0.7$ and $b = 0.3$ for the discussed example.

2.2.4 Element probability defect

The previous calculated probabilities only account for the exact target structure in the ensemble of structures. However, the aptamer or `ac` structural element is also present (and thus functional) in many other structures that include additional base pairs or have less base pairs, e. g., in the MFE structure shown above. To be correct, we should include all structures of the ensemble that do not conflict with our structural element in the probability calculations.

Therefore, we use the hard constraints framework of the `ViennaRNA package` [9] to calculate the ensemble free energy and thus the partition function only of structures containing our element. With that we can calculate the probability of our structural element in the ensemble analogous to the previously shown equations.

Hard constraints restrict the conformations of RNA structures to states containing a combination of unconstrained "." bases, bases that have to be unpaired "x", bases that have to be paired no matter to which binding partner "|" and base pairs indicated by matching brackets "( )". It is furthermore possible to specify if a specific base has to be paired with a binding partner up- or downstream by using the symbols “<” and “>”, respectively.

```
$ cat ac.txt | RNAfold -C -p --canonicalBPonly
  AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCAGUUGUUGAGGGGGCUCAAUGAC
  ................((((((((.(((((((((((......))))))))))).).)))).))) (-21.60)
  ....{,,......,,,((((((((.(((((((((((......))))))))))).).)))).))) [-22.38]
  ................((((((((.(((((((((((......))))))))))).).)))).))) {-21.60 d=2.28}
 frequency of mfe structure in ensemble 0.284221; ensemble diversity 3.59
```

This performs a constrained (`-C`) partition function (`-p`) folding. The `--canonicalBPonly` option removes non-canonical base pairs, e. g., U-U, from the structure constrain if they where erroneously added. Re-running the last command without the `-C` option yields the ensemble free energy of the complete ensemble.

$$P(x \mid \Phi_{\mathrm{ac}}) = \exp(-\frac{G(x|\Phi_{\mathrm{ac}}) - G(x|\Phi)}{RT}) = \exp(\frac{22.38 - 22.75}{0,61632}) = 0.5486$$

$$P(x \mid \Phi_{\mathrm{bc}}) = \exp(\frac{21.87 - 22.75}{0,61632}) = 0.2398$$

To gain $P(x \mid \Phi_{\mathrm{lc}})$, we can again use the `--motif` option to add the ligand binding energy contribution to states with the binding pocket.

$$P(x \mid \Phi_{\mathrm{lc}}) = \exp(\frac{30.98 - 30.98}{0,61632}) = 1.0$$

The final objective function now is:

$$f(x) = P(x \mid \Phi_{\mathtt{lc}}) \cdot (1 - |a - P(x \mid \Phi_{\mathtt{ac}})|) \cdot (1 - |b - P(x \mid \Phi_{\mathtt{bc}})|) \qquad (2)$$

2.3 Combining sequence sampling, objective and optimization method

There are many RNA design approaches available and most of them use a predefined optimization procedure [1]. With our recently published `RNAblueprint` approach [3] we decoupled the sampling of sequences compatible to one or more structural constraints and the subsequent optimization to score sequences and to select most promising ones. The sampling is implemented in `C++` while the optimization procedure can be carried out with the `Python` or `Perl` interface by each user individually. A `Python` module called `RNAsketch` contains essential features such as various optimization algorithms, common energy calculations and programs for various design applications.

The `ligandswitch.py` script implements a simple gradient walk optimization approach, the presented design objectives and uses `RNAblueprint`for constraint sequence sampling.

```
$ python ligandswitch.py mfe       # mfe defect
$ python ligandswitch.py prob      # probability defect
$ python ligandswitch.py           # element probability defect
```

Alternatively, you can use the `design-ligandswitch.py` program from the `RNAsketch` module.

```
$ echo -e "(((((...((((((((.....)))))...)))...))))).......................
  .........................(((((((((((......)))))))))))...........
  AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCANNNNNNNNNNNNNNNNNNNNNN"\
   | design-ligandswitch.py -r 70:30 --ligand \
    "GAUACCAG&CCCUUGGCAGC;(...((((&)...)))...);-9.22"
```

In the following we will discuss how the generated sequences can be analyzed *in silico* to validate their functionality.

3 *In silico* analysis of the candidates

During the design process only thermodynamic parameters were used to control the optimization. However, we assume that kinetic processes will drive the implemented switching dynamics. Thus, we will describe how to estimate kinetic features of designed sequences in the following. The partition function fold was used to estimate certain probabilities or energies of states and sub-structures. To kinetically analyze the system, the complete structural ensemble needs to be generated. `RNAsubopt` can be applied to create all structures a given sequence can adopt within a specified energy band above the MFE.

```
$ cat testing_example.txt | RNAsubopt -e 1.2 -s
  AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCAGUUGUUGAGGGGGCUCAAUGAC -21.60   1.20
  ................((((((((.(((((((((((......))))))))))).).)))).))) -21.60
 ....(((......)))((((((((.(((((((((((......))))))))))).).)))).))) -21.50
  (((((...((((((((.....)))))...)))...)))))..(((((((((....))))))))) -21.10
  (((((...(((((((((...))))))...)))...)))))..(((((((((....))))))))) -20.80
  .((((...((((((((.....)))))...)))...))))...(((((((((....))))))))) -20.80
  ..(((.......))).((((((((.(((((((((((......))))))))))).).)))).))) -20.60
  .((((...(((((((((...))))))...)))...))))...(((((((((....))))))))) -20.50
  ..((.......))...((((((((.(((((((((((......))))))))))).).)))).))) -20.50
  ...(((..((.(((((.....)))))((((((((((......))))))))))))..))).....  -20.40
```

The above system call generates all sub-optimal structures 1.2 *kcal*/*mol* above the ground state energy of a designed example sequence. Note, that the number of generated structures grows exponentially with both sequence length and the selected energy band. Not only CPU time is consumed but also files with hundreds of gigabytes in size can be the result. This already indicates the necessity to coarse grain the high dimensional structure landscape an RNA sequence spans. The `barriers` program implements a flooding algorithm and reduces the landscape to a selected number of local minima. Those are pairwise connected by so called saddle points. In order to get a connected barrier tree two conditions need to be fulfilled: i) the `RNAsubopt` output has to be sorted and ii) the energy band has to be large enough to connect all local minima. The first is mandatory to run `barriers`. As the sorting routine applied by `RNAsubopt` (`-s` option) might fail on huge inputs even on high memory machines with hundreds of giga- or even terabytes of RAM, a workaround is to pipe the `RNAsubopt` output to Unix's `sort`. The following system call produces the same output as

the one before but scales with the memory consumption of the possibly huge ensemble size.
Attention! You need 20GB of RAM to run this command!

```
$ cat testing_example.txt | RNAsubopt -e 22.60 | sort -k2,2n -k1,1r -S20G > example.sub
```

The main memory buffer is set to be 20GB here. Above this threshold `sort` will dump data to temporary files on the hard drive which implies performance loss but makes it still possible to process corresponding sequences. For each of the screened design candidates we estimated the energy band by folding the sequences with `RNAfold` and taking `-e` $(-1 \times (\mathrm{MFE}+1))$ to convert the energy into a positive value and taking a few more structures with positive energy values into account. For our examples this was sufficient to fulfill condition ii) in order to generate connected barrier trees.
To reduce the number of generated sub-optimal sequences it is possible to apply the `--noLP` option to `RNAsubopt`. This skips all structures containing lonely pairs, i.e. helices of length one, therefore reducing file size from 16GB to 459MB for the given example. Note, that also `barriers` has to be run subsequently with `-G RNA-noLP`. However, for the shown example the predicted MFE structure would dramatically change this way. The previous ground state containing the alternative structural element would only be the third stable state now while the MFE structure contains the binding competent aptamer. This, of course, has a large impact on the predicted kinetics, see Figure 2. Depending on the research question `--noLP` results can give valuable insights of the studied system and dramatically speed up the screening process. Nevertheless, we recommend to rerun the analysis for promising candidates with the full `RNAsubopt` output, especially if low energy structures contain lonely pairs. The following example generates the full `RNAsubopt` output and runs `barriers`. Keep in mind that this call may take some time as the `RNAsubopt` output is 16GB in size which is sorted afterwards.
Attention! You need 20GB of RAM to run this command!

```
$ barriers --max=500 -G RNA --bsize --rates < example.sub > example.bar
```

The `--max=500` option specifies the number of minima to be generated, `-G` graph type is set to RNA, and `--bsize` and `--rates` enable to print the size of each basin corresponding to a minimum and to compute rates between these macro states. The output is saved in the `example.bar` file, a graphical representation of the barrier tree is by default saved in the post-script format to a file named `tree.ps`. Transition rates from each macro state (basin) to all others are stored as a matrix in `rates.bin` and `rates.out` The latter is needed to run `treekin` (version 0.3.1):

```
$ treekin --p0 1=1 -m I --t8=1E12 -f rates.out < example.bar > example.tkin
```

where `-m I` tells `treekin` to parse the rates file specified by `-f` as `barriers` output, `--t8` sets the maximum simulation time to 1E12 arbitrary time units (AU) and `--p0` sets the initial population size of the selected minimum of the

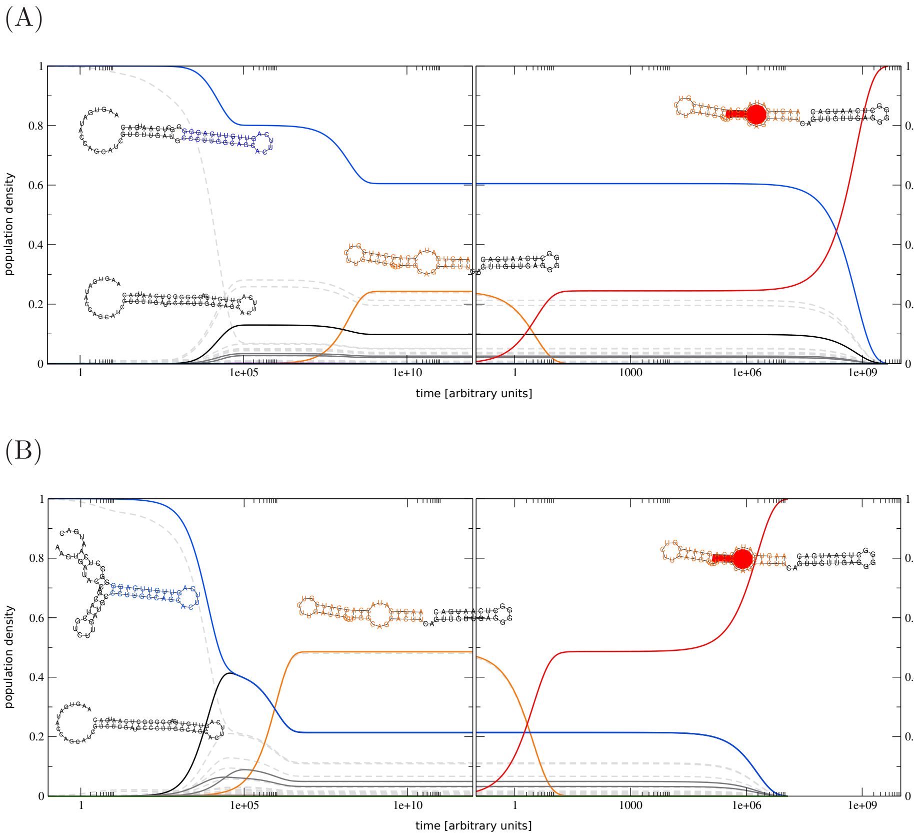


Figure 2: Simulated kinetics using (A) the complete and (B) a reduced structure ensemble by avoiding lonely pairs (using `--noLP`). In both cases the simulation is started with the complete population in a structural state that contains the alternative structural element and is present in both, the complete and the reduced structure ensemble. The left part of each plot shows the dynamics until the system is equilibrated. Whereas the right part depicts the simulated systems kinetics after ligand addition. Dashed gray lines indicate the systems kinetics without coarse graining. By design, the population density in equilibrium of all structures containing alternative and the binding competent structural element should be 0.7 and 0.3, respectively. Colored lines display the coarse grained kinetics where states containing specific structural elements are merged. For most prominent states the corresponding secondary structures of the most stable representative are shown using the same color. The blue and the black curves are not merged as the perfectly stacked stem is only 6 nt in the latter structure and was forced to be at least 10 nt when adding up states that correspond to the alternative conformation.

barrier tree. Here we set the global minimum of the barrier tree to be 100%. The output can be visualized using `xmgrace` and the following system call:

```
$ xmgrace -log x -nxy example.tkin
```

This plots 500 independent curves, one for each minimum of the barrier tree. However, we optimized for sub-structures in the ensemble and the corresponding structural conformation and not for a specific state. Thus, we want to collect states that exhibit our structural features, i.e. ligand binding stem or alternative stem into combined density curves. We implemented a `Perl` script called `coarsify_bmap.pl`[1] that does the job and can be applied to `example.bar` and `example.tkin` output as follows:

```
coarsify_regex.txt

# ?25(((((((((((......))))))))))) | ?26((((((((((......))))))))))
^.{25}\({11}\.{6}\){11}[\.\(\)]{11}|^.{26}\({10}\.{6}\){10}[\.\(\)]{11}

# ?2(((...((((((((.....)))))...)))...))) | ?2(((...((((((((.....))))...))))...)))"
^.{2}\({3}\.{3}\({8}\.{5}\){5}\.{3}\){3}\.{3}\){3}|^.{2}\({3}\.{3}\({8}\.{5}\){4}\.{3}\){4}\.{3}\){3}

$ perl coarsify_bmap.pl -regs coarsify_regex.txt -minh 30 --tkin example.tkin --outdir coarse example.bar
```

`coarsify_bmap.pl` merges local minima of a given barrier tree in two ways: (i) if the barrier height of a minimum is below the selected `-minh` value it is merged to its neighbor and the population density of this neighbor is increased accordingly and (ii) if local minima contain similar structural elements, specified as regular expressions (`coarsify_regex.txt`), they are merged. Note that minima containing a different set of these structural elements are never merged although (i) would be applicable. For the above example all minima are merged as `-minh` is larger than the energy band generated by `RNAsubopt`. However, the two specified regular expressions combine minima that are compatible with the initial structural constraints of the design and keep the remaining landscape separate. The coarse grained `barrier` and `treekin` output is written to `coarse/example.bar` and `coarse/example.tkin`, respectively.
Visualizing the coarse grained `treekin` output approximately shows the expected population density of the two designed structural conformations, see Figure 2. This way we verified that the partition function estimate used during optimization matches the results of the kinetic simulation if no ligand is present. When sketching the design, Figure 1, we assume that an RNA:ligand complex has a rather low k_{off} compared to k_{on} rate. For the theophylline aptamer these values are in accordance to published rates of $0.07 \pm 0.02 s^{-1}$ and $(1.7 \pm 0.2) \times 10^5 M^{-1} s^{-1}$ at 25°C, respectively [7]. Therefore, as soon as the the RNA is in its thermodynamic equilibrium, the effect of ligand addition can be simulated by starting `treekin` (version 0.3.1) with the population density of the last time point in `example.tkin` and making the binding competent state absorbing, i.e. setting `-a` option to the most stable binding competent state.

[1] `https://github.com/ViennaRNA/BarMap/`

```
$ grep -v "#" example.tkin | tail -n 1 | \
 perl -ae '{for($i=1; $i<scalar(@F); $i++){print "--p0 ",$i,"=",$F[$i]," "}}' > states
$ treekin -m I `cat states` -f rates.out --t8=1E12 -a 3 < example.bar > example_absorb.tkin
$ coarsify_bmap.pl -regs coarsify_regex.txt -minh 30 --tkin example_absorb.tkin --outdir coarse example.bar
$ rm states
```

First, the last time point in `example.tkin` is extracted and converted such that the output saved in `states` can be used as repeated `--p0` parameter of `treekin`. Second `treekin` is called and its output is stored in `coarse/example_absorb.tkin` which is subsequently coarse grained. Finally the temporary file `states` is removed. Visualization of the coarse grained absorbing landscape with

```
$ xmgrace -log x -nxy coarse/example_absorb.tkin
```

shows a rather slow switching behavior upon ligand addition, see Figure 2. After about $2.89+08$ AU, which can be approximately mapped to 24 min [13], 50% of the RNA molecules are in the ligand bound state. We attribute this rather slow refolding process to the 70:30 ratio of the alternative and the binding competent conformation. Changing this to 50:50 speeds up the refolding dramatically but removes the feature of the structural state without ligand to dominate the ensemble. If the described design process is extended to design translational or transcriptional riboswitches one would of course like to also have well defined states for the expression platforms. All herein described ideas and sequences are so far of theoretical nature only. Please note that experimental validation of the mechanistic details is not a straight forward task.

References

[1] S. Findeiß, M. Wachsmuth, M. Mörl, and P. F. Stadler. Design of transcription regulating riboswitches. *Methods Enzymol*, 550:1–22, 2015.

[2] C. Flamm, I. L. Hofacker, P. F. Stadler, and M. T. Wolfinger. Barrier trees of degenerate landscapes. *Zeitschrift für Physikalische Chemie*, 216(2/2002), jan 2002.

[3] S. Hammer, B. Tschiatschek, C. Flamm, I. L. Hofacker, and S. Findeiß. RNAblueprint: Flexible multiple target nucleic acid sequence design. *Bioinformatics*, 33(18):2850–2858, Sept. 2017.

[4] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[5] C. Höner zu Siederdissen, S. Hammer, I. Abfalter, I. L. Hofacker, C. Flamm, and P. F. Stadler. Computational design of RNAs with complex energy landscapes. *Biopolymers*, 99:1124–1136, 2013.

[6] R. D. Jenison, S. C. Gill, A. Pardi, and B. Polisky. High-resolution molecular discrimination by RNA. *Science*, 263(5152):1425–1429, Mar 1994.

[7] F. M. Jucker, R. M. Phillips, S. A. McCallum, and A. Pardi. Role of a heterogeneous free state in the formation of a specific RNA-theophylline complex. *Biochemistry*, 42(9):2560–2567, Mar 2003.

[8] R. Lorenz, S. H. Bernhart, C. H. z. Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. ViennaRNA Package 2.0. *Algorithms for Molecular Biology*, 6(1):26, Nov. 2011.

[9] R. Lorenz, I. L. Hofacker, and P. F. Stadler. RNA folding with hard and soft constraints. *Algorithms for Molecular Biology*, 11:8, 2016.

[10] R. Lorenz, D. Luntzer, I. L. Hofacker, P. F. Stadler, and M. T. Wolfinger. SHAPE directed RNA folding. *Bioinformatics*, 32(1):145–147, Jan 2016.

[11] D. H. Mathews, M. D. Disney, J. L. Childs, S. J. Schroeder, M. Zuker, and D. H. Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc Natl Acad Sci U S A*, 101(19):7287–7292, 2004.

[12] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953.

[13] B. Sauerwine and M. Widom. Folding kinetics of riboswitch transcriptional terminators and sequesterers. *Entropy*, 15(8):3088–3099, jul 2013.

[14] A. Taneda. Multi-objective optimization for RNA design with multiple target secondary structures. *BMC Bioinformatics*, 16(1):280, Sept. 2015.

[15] D. H. Turner and D. H. Mathews. NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic Acids Res*, 38(Database issue):D280–D282, 2010.

[16] M. T. Wolfinger, W. A. Svrcek-Seiler, C. Flamm, I. L. Hofacker, and P. F. Stadler. Efficient computation of RNA folding dynamics. *Journal of Physics A: Mathematical and General*, 37(17):4731–4741, apr 2004.

	Software	Description	Ref	URL
RNA related	`RNAblueprint` v1.2	Fair sampling approach that generates sequences compatible to sequence constraints and to one or more structural constraints. You need to install the boost library first.	[3]	`https://github.com/ribonets/RNAblueprint`
	`barriers v1.7.0`	Generates a coarse grained energy landscape given a energy sorted list of sub-optimal RNA secondary structures.	[2]	http://www.tbi.univie.ac.at/RNA/Barriers/
	`treekin v0.3.1`	Calculates folding kinetics on a coarse grained energy landscape. One problem that often occurs during treekin installation is its dependency on `blas` and `lapack` packages. Try to install them first. Note that the `-a` option is broken in v0.4	[16]	http://www.tbi.univie.ac.at/RNA/Treekin/
	`ViennaRNA` package v2.4.11		[8]	`http://www.tbi.univie.ac.at/RNA/`
	`RNAfold`	Calculates minimum free energy secondary structures and partition function of nucleic acid sequences.		
	`RNAdistance`	Given two secondary structures this program calculates their dissimilarity.		
	`RNAsubopt`	Calculates sub-optimal secondary structures a nucleic acid sequence can fold into.		
Other	`sort`	As part of the gnu core utils this program takes a text file and sorts it in the specified order		http://www.gnu.org/software/coreutils/sort
	`xmgrace`	`xmgrace` is a full-featured graphical user interface of `grace` to make two-dimensional plots.		http://plasma-gate.weizmann.ac.il/Grace/

Table 1: Summary of the software used throughout the presented protocol. RNA related software tools are either standalone or part of the `ViennaRNA package`. How they can be installed is documented on the web pages listed. Standard Unix tools are tagged as Other and are typically easy to install with the package manager of any distribution.